GNU/Linux

Sommaire

GNU/Linux

Présentation

Linux est le nom couramment donné à tout **système d'exploitation** (*operating system*) libre fonctionnant avec le noyau Linux. C'est une implémentation libre du système **UNIX** respectant les spécifications **POSIX** (normes techniques de l'IEEE).

Remarque : un système d'exploitation est une couche logicielle (software) qui permet et coordonne l'utilisation du matériel (hardware) entre les différents programmes d'application.

GNU/Linux est le nom parfois donné à un système d'exploitation associant des éléments essentiels (*shell*, compilateurs, bibliothèques C, commandes, etc ...) du projet **GNU** (*GNU's Not UNIX*) et d'un noyau (*kernel*) Linux. C'est une terminologie créée par le projet Debian et reprise notamment par **Richard Stallman**, à l'origine du projet de travail collaboratif GNU et de la licence libre **GPL** (*General Public Licence*).

Par exemple : Android est un système basé sur Linux mais pas sur GNU.

Le noyau Linux a été initialement écrit par **Linus Torvalds**, un étudiant finlandais au début des années 90. Depuis, des centaines de développeurs et des entreprises de toutes tailles participent au projet, dont Linus Torvalds est toujours le coordinateur.

Le système avec les applications est le plus souvent distribué sous la forme de **distributions Linux** comme Slackware, Debian, Red Hat, Mandriva ou **Ubuntu** ...

La différence essentielle de Linux par rapport à d'autres systèmes d'exploitation concurrents (comme Mac OS, Microsoft Windows et Solaris) est d'être un système d'exploitation libre, apportant quatre libertés aux utilisateurs, définies par la licence GNU GPL, les rendant indépendants de tout éditeur et encourageant l'entraide et le partage :

- « utiliser le logiciel sans restriction »
- « étudier le logiciel »
- « modifier pour l'adapter à ses besoins »
- « redistribuer sous certaines conditions précises »

Remarque : Un logiciel libre n'est pas nécessairement gratuit, et inversement un logiciel gratuit n'est pas forcément libre.

Notion de système d'exploitation

De manière générale, un système d'exploitation :
- permet l'exploitation des périphériques matériels dont il coordonne et optimise l'utilisation ;
- propose aux logiciels applicatifs des interfaces de programmation standardisées qui simplifient l'utilisation des matériels et des services qu'il offre ;
- coordonne l'utilisation du ou des processeur(s), et accorde un certain temps pour l'exécution de chaque processus (multi-tâche) ;
- gère l'espace mémoire pour les besoins des programmes ;
- organise le contenu des disques durs ou d'autres mémoires de masse en fichiers et répertoires ;
- fournit les interfaces homme-machine des différents programmes ;
- réalise enfin différentes fonctions visant à assurer la fiabilité (tolérance aux pannes, isolation des fautes) et la sécurité informatique (traçabilité, confidentialité, intégrité et disponibilité).

Philosophie UNIX

« L'univers a 40 ans »

UNIX a marqué à jamais l'histoire de l'informatique et continue à le faire, ceci pour une raison très simple : derrière cette famille de systèmes, il y a une idée ou plutôt un ensemble d'idées et de préceptes. Derrière UNIX, il y a une philosophie qui sert de ligne de conduite et de fil d'Ariane. Comprendre cette philosophie et la respecter le mieux possible assure une stabilité et une pérennité sans précédent.

Résumer la philosophie d'UNIX n'est pas chose évidente. Il s'agit d'un ensemble de principes. Nombreux sont ceux qui ont essayé de les résumer ou les lister (taper « philisophie UNIX » ou « *less is more* » dans un moteur de recherche).

Des programmes qui effectuent une seule chose et qui le font bien

Voilà la base de toutes choses dans le monde UNIX (« dans le monde » tout court peut-être également).

Le silence est d'or

En d'autres termes, lorsqu'un programme n'a rien à dire, il doit garder le silence. Ce n'est que lorsqu'il y a un problème qu'un outil doit devenir bavard et signaler explicitement une erreur. Un programme qui fait ce qu'on lui demande n'affiche rien, ne signale rien. C'est le cas de la plupart des outils de base en ligne de commande.

Des programmes qui collaborent

Si tous les programmes ne font, chacun, qu'une chose et qu'ils la font bien, ceci implique qu'ils doivent alors fonctionner de concert pour pouvoir achever des tâches plus importantes. Ces « briques » doivent alors collaborer les unes avec les autres du mieux possible. Le but est de former un système complet où la somme des parties est supérieure à l'ensemble. Lorsqu'on dispose d'un ensemble de briques fiables, il est possible de construire un mur solide.

Des programmes pour gérer des flux de texte

Les flux de texte représentent une interface universelle (la seule ?). La notion de flux de texte est véritablement caractéristique des UNIX.

Citations

> « Unix est convivial. Cependant Unix ne précise pas vraiment avec qui. » Steven King
>
> « Unix ne dit jamais 's'il vous plaît'. » Rob Pike
>
> « Unix est simple. Il faut juste être un génie pour comprendre sa simplicité. » Denis Ritchie
>
> « Unix n'a pas été conçu pour empêcher ses utilisateurs de commettre des actes stupides, car cela les empêcherait aussi des actes ingénieux. » Doug Gwyn

Conclusion

Si je devais répondre à la question « Qu'est-ce qu'un UNIX ? », je répondrais par ce type de commande (pleine de magie et d'intelligence) :

```
$ history | grep -v " h" | sed 's/[ \t]*$//' | sort -k 2 -r | uniq -f 1 | sort -n
```

[Extrait d'un article de Denis Bodor dans GNU/Linux Magazine HS n°46]

Manipuler sous Linux

L'interface homme-machine (IHM)

L'interface homme-machine (**IHM**) permet à un utilisateur de dialoguer avec la machine.

On distingue deux types d'IHM :
- **GUI** (*Graphical User Interface*) ou « interface utilisateur graphique » : les parties les plus typiques de ce type d'environnement sont le pointeur de souris, les fenêtres, le bureau, les icônes, les boutons, les menus, les barres de défilement, ... Les systèmes d'exploitation grand public (Windows, MacOS, GNU/Linux, etc.) sont pourvus d'une interface graphique qui, dans un soucis d'ergonomie, se veut conviviale, simple d'utilisation et accessible au plus grand nombre pour l'usage d'un ordinateur personnel.
- **CLI** (*Command Line Interface*) ou « interface en ligne de commande » est encore utilisée en raison de sa puissance, de sa grande rapidité, son uniformité, sa stabilité et du peu de ressources nécessaires à son fonctionnement. Le système d'exploitation permet cette possibilité par l'intermédiaire d'un interpréteur de commandes (le *shell*). Beaucoup de serveurs ne s'administrent qu'en ligne de commande.

Conventions

Tous les exemples d'exécution des commandes sont précédés d'une **invite** utilisateur ou ***prompt*** spécifique au niveau des droits utilisateurs nécessaires sur le système :

- toute commande précédée de l'invite `$` ne nécessite aucun privilège particulier et peut être utilisée au niveau utilisateur simple ;
- toute commande précédée de l'invite `#` nécessite les privilèges du super-utilisateur (*root*).

Évidemment, il ne faudra jamais taper l'invite (`$` ou `#`) lorsque vous testerez par vous même les commandes indiquées.

Conseils

Travailler toujours en mode « plein écran ».

N'utilisez pas la souris (ou très peu). Il existe beaucoup de raccourcis clavier et de touches « magiques » :
- la touche **tabulation**  (la plus utile) permet la complétion en ligne de commande. Le *shell* effectue la complétion en considérant successivement le texte comme une variable (s'il commence par `$`), un nom d'utilisateur (s'il commence par `~`), un nom d'hôte (s'il commence par `@`), ou une commande (y compris les alias et les fonctions). Si rien ne fonctionne, il essaye la complétion en nom de fichier.
- Les touches flèches  et  servent à parcourir l'historique des commandes déjà saisies.

Utiliser plusieurs sessions *shell* (ou onglets ou fenêtres) en parallèle. PAr exemple, vous en utiliserez une pour saisir vos commandes et l'autre pour consulter les indispensables pages de manuel. Pour basculer de l'une à l'autre :
– en mode console : Ctrl + Fx (où x est un chiffre identifiant le terminal)
– en mode graphique, avec 2 onglets : Ctrl + Page ↑ ou Ctrl + Page ↓ , Shift ⇑ + ← ou Shift ⇑ + →
– en mode graphique, avec 2 fenêtres : Alt + ⟷

Structure d'une commande

Une **commande** Unix est un ensemble de mots séparés par des **espaces**. Les caractères espace et tabulation sont interprétés comme des séparateurs par le *shell* (voir la variable `IFS`). La syntaxe d'une commande est la suivante :

```
$ commande [options] <parametres>
```

Le premier mot est le nom de la commande. Les autres mots sont des paramètres (ou arguments) de la commande. Certains mots sont des options qui changent le comportement de la commande. Les 2 crochets « `[` » et « `]` » indiquent que les options ne sont pas obligatoires. Il ne faut pas taper ces crochets sur la ligne de commande.

Avant, une option était introduite par le signe « `-` » suivi d'une seule lettre. Le standard actuel GNU pour les options est d'utiliser « `--` » suivi du nom de l'option pour des raisons de clarté et de portabilité. L'ordre des options n'a pas souvent d'importance :

```
$ ls --all -l --si     ou     $ ls -l --si --all
$ ls -l $HOME/tmp
```

Différents types de commande

Il existe plusieurs type de commandes :
– les **commandes internes** (au *shell*) : comme `history`, `test`, ...
– les **commandes externes** (donc des programmes) : comme `ls`, `mkdir`, ...
– les ***alias*** (voir plus loin) : comme `ll`, ...

Les commandes externes (donc des exécutables) sont généralement stockées dans un répertoire de nom `bin`. Il existe des exécutables dans :

– le répertoire `/sbin` : les commandes pour *root* (l'administrateur)
– le répertoire `/bin` : des commandes et des *shells*
– le répertoire `/usr/bin` : le répertoire de base des programmes

Remarque : comme le système ne connaît pas les endroits où vous placez vos programmes, il faudra lui indiquer dans la variable d'environnement `$PATH`.

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

$ type echo
echo est une primitive du shell

$ type strings
strings est /usr/bin/strings
```

Ceci est un exemple, cliquez sur le lien de téléchargement pour obtenir le cours complet.