GNU/Linux

- Objet : Système GNU/Linux
- Niveau requis :
  débutant, avisé
- Commentaires : *Informations concernant le système GNU/Linux.*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !
- Suivi :
  - Création par smolski le 02/05/2013
  - Testé par .... le ....
- Commentaires sur le forum : C'est ici[1)]

Le PC

Un PC est une machine qui écrit des 0 et des 1 sur les secteurs d'un disque et qui utilise une carte graphique pour nous permettre de les lire sur un écran.

Système d'exploitation

Un système d'exploitation (ou **OS**) est un logiciel qui, en première couche, possède les bases de fonctionnement pour gérer le matériel installé[2)].

L'OS fournit aussi *l'interface graphique* qui se trouvera être plus ou moins séparée de cette première couche.

Par exemple. Windows mélange ces deux couches alors que, dans le but d'un fonctionnement et d'une clarté meilleurs pour l'utilisateur, GNU/Linux les sépare davantage.

Interface graphique

Bien que fournie avec l'OS, l'interface graphique n'en fait pas réellement partie.

L'interface graphique vient se placer au-dessus de ce dernier et se divise en plusieurs parties :

1. Le *X.Org* qui se charge de tout qui concerne le matériel (par exemple la carte graphique : *affichage*, *manipulation* de ce qui apparaît à l'écran, gestion de la *HD* et de la *3D*...).
2. Gestionnaires de fenêtres et environnement de bureau.

La mémoire

C'est l'OS qui gère la mémoire.

La mémorisation permet de conserver et de distinguer les données propres à chacune des

applications actives dans la machine.

La mémoire, c'est d'abord la **RAM** installée[3], mais c'est aussi la partition swap créée sur le disque dur, lors de l'installation, et qui est utilisée par l'OS en cas de besoin.

Détail

En effet, une application en cours a besoin de conserver en mémoire des données à sauvegarder le temps de son exécution.

Plusieurs applications[4] peuvent demander à s'exécuter en même temps et doivent donc *chacune* conserver en *mémoire* des données temporaires différentes.

Par exemple l'application gimp conservera une couleur particulière à reproduire pendant qu'une application de traitement de texte conservera une police de texte en cours et qu'un navigateur visitera une page sur la *Toile*...

1. L'OS se charge d'écrire ces mémorisations *directement* dans des zones séparées de la mémoire.
2. Les programmes ne voient qu'un espace de mémoire *virtuelle* dédié par l'OS où ils stockent les données qu'ils ont besoin de conserver pour fonctionner.

Quantification de la mémoire

Il n'y a pas plus de *norme standard* que de *norme Microsoft* pour la quantification de l'espace mémoire.
Par contre il y a beaucoup de confusion dans ce domaine.

Pour rappel, il y a deux familles de préfixes multiplicateurs normalisés :

1. les préfixe SI (ou décimaux) qui représentent des *puissances de 10* : k=1000, M=1000^2, G=1000^3, T=1000^4, etc.
2. les préfixes binaires qui représentent des puissances de 2 (1024=2^10) : Ki=1024, Mi=1024^2, Gi=1024^3, Ti=1024^4, etc.

Comme on peut le voir, les noms, symboles et valeurs des préfixes binaires et décimaux sont proches, mais leurs valeurs s'écartent en augmentant :

- 2 % entre k et Ki,
- 5 % entre M et Mi,
- 7 % entre G et Gi,
- 10 % entre T et Ti.

Les préfixes multiplicateurs binaires sont plus pratiques pour noter des valeurs qui sont des puissances entières de 2, comme la capacité des composants de mémoire (RAM ou flash) ou la taille d'un espace d'adressage, c'est pourquoi ils sont largement utilisés en électronique et informatique (mais pas en réseau et télécom où on utilise les préfixes décimaux).

Malheureusement la normalisation des préfixes binaires ci-dessus a été tardive, et entre-temps un usage *sauvage* des symboles des préfixes SI, mais en leur attribuant des valeurs binaires, s'est

développé dans ces domaines, et les mauvaises habitudes ont la vie dure.

Windows est dans ce cas : il utilise de façon constante la notation des symboles préfixes SI mais en leur donnant des valeurs binaires.
Même chose pour le JEDEC, l'organisme de normalisation qui regroupe notamment les fabricants de composants électroniques de mémoire.

En ce qui concerne GNU/Linux, le noyau, qui gère la mémoire, fait plutôt la même chose. En revanche dans les programmes utilisateur, certains continuent à faire pareil, tandis que d'autres utilisent désormais les notations normalisées de façon correcte (ifconfig, parted).
free n'affiche pas de préfixe explicitement, mais affiche les valeurs en kibi, mébi ou gibi-octets, alors que sa page de manuel et son aide en ligne parlent de kilo, méga ou giga-octets.

Dans tous les cas, si vous avez un doute sur la signification d'un préfixe multiplicateur, c'est simple : affichez la valeur exacte sans préfixe, en bits ou octets et vous verrez bien.

Pour en revenir au sujet, la taille d'une barrette de RAM est une puissance entière de 2 dont la valeur est légèrement supérieure à la puissance entière de 10 correspondante.

Une barrette marquée **4 GB** par le fabricant fait en réalité 4 Gio, soit 4,3 Go. Pas 3,8 Gio ni 3,8 Go.

Ne pas oublier qu'une partie de la mémoire est réservée par la carte mère (pour le GPU intégré notamment), et une autre est réservée par le noyau lui-même. Le *total* affiché par **free** ne compte pas la mémoire réservée par le noyau. Pour avoir une idée de ces quantités, on peut rechercher dans les logs du noyau une ligne qui commence par *Memory* :

```
dmesg |grep Memory
```

```
Memory: 126248k/131008k available (1762k kernel code, 4244k reserved, 853k data, 252k init, 0k highmem)
```

Il s'agit d'une machine qui a 128 Mio de RAM, soit 128*1024=131072 Kio, valeur très proche de 131008 k. 126248 k est plus proche de la valeur qui est affichée par free dans la colonne *total* (126564).

Je n'ai jamais vu de chipset qui limite la RAM à 3 Gio (ou 3 Go, ce qui a encore moins de sens). 2 Gio, 4 Gio, la limite est toujours une puissance entière de 2, ce qui est logique.

Merci **raleur**.

Lien des sources sur le forum df : https://debian-facile.org/viewtopic.php?pid=157275#p157275

Nota

Pour apporter des commentaires, utilisez comme pour tous les tutos le lien de l'en-tête et non pas ce dernier lien qui est une résolution particulière et sert à remettre le contexte ainsi qu'aux remerciements mérités à qui de droit.

Multitâche

Voir à propos des processeurs ce post dans le forum :

- https://debian-facile.org/viewtopic.php?id=16477

Schématiquement, un PC standard d'un seul processeur ne peut exécuter qu'une opération à la fois. Pourtant, nous pouvons simultanément :

1. utiliser une application,
2. naviguer sur internet,
3. copier des fichiers,
4. etc...

Et nous le faisons comme si tous ces programmes s'exécutaient ensemble !
C'est le principe du multitâche dit *préemptif* [5].

En fait, l'OS répartit ces actions en les exécutant non chacune dans son ensemble, mais chacune en pointillé avec les autres, et il le fait si rapidement qu'il donne l'illusion qu'elles s'exécutent toutes en même temps !

En fait, entre ces pointillés il se crée un temps de quelques millisecondes (un *time-slice*) variable selon le système installé.

Toutefois, le passage d'une application à l'autre a un coût de temps qui est d'autant plus long qu'il s'y ajoute la restauration de l'ensemble des paramètres de chaque application en exécution simultanée, notamment la reconnaissance de la mémoire virtuelle attribuée et l'instant de calcul du processeur avant l'interruption.

Nota

1. Plus le **time-slice** est faible, *plus l'exécution du multitâche* ne sera pas apparente pour l'utilisateur.
2. C'est l'OS[6] qui dirige le multitâche et non l'application.
3. L'application n'utilise directement que la RAM.

Le pilote de périphérique

Le pilote de périphérique[7] permet à l'OS de faire fonctionner le matériel installé via une interface toujours identique (*Hardware Abstraction Layer*) malgré leurs composants électroniques différents.

Par exemple, les cartes graphiques diffèrent les unes des autres. Le driver en établit les spécifications techniques et permet à l'OS de demander d'afficher les données sur l'écran toujours selon la même méthode.
Ainsi, quels que soient les composants des cartes graphiques, pour l'OS un point reste un point, un rectangle un rectangle, etc...

Des normes permettent de standardiser un peu tout ça. Par exemple, la norme **VGA** pour les cartes graphiques. Cependant, pour la HD ou la 3D chaque fabricant crée ses propres spécifités.

Les drivers font partie de l'OS bien qu'ils puissent être fournis séparément par le constructeur.

Mode

Aujourd'hui, il existe un mode dit *protégé* qui permet d'avoir entre autres un meilleur modèle d'accès à la mémoire.
Dans ce dernier, il y a plusieurs niveaux d'exécution qui permettent d'avoir des privilèges différents :

1. le mode noyau, (appelé ring 0),
2. un mode intermédiaire (ring 1), et
3. le mode utilisateur (ring 2).

GNU/Linux utilise uniquement le ring 0 (pour le noyau...) et le ring 2 pour les applications.

firmware

Les Firmware sont des *micro-programmes* souvent fournis par le constructeur que le pilote noyau doit charger dans une RAM incluse dans le matériel devant la RAM du PC gérée elle par l'OS.
Ce matériel peut aussi posséder un processeur agissant préalablement à celui du PC géré par l'OS.

De nombreux périphériques nécessitent aujourd'hui que soient chargés des *firmware*, et comme ceux-ci sont rarement libres, ils ne sont pas installés par défaut sous Debian.

Démarrage

Au démarrage, **GNU/Linux** doit passer par plusieurs *initialisations* et divers programmes pour devenir utilisable
Chacune de ces étapes peut être configurée afin d'obtenir, par exemple, des fonctions supplémentaires, gagner du temps au démarrage, etc...

L'initialisation, c'est quand la racine est montée et le processus /sbin/init est lancé.

Rappel de la séquence de démarrage normale :

1. GRUB charge le noyau (/boot/vmlinuz) et l'initramfs (/boot/initrd.img).
2. GRUB lance le noyau.

3. Le noyau monte l'initramfs comme racine initiale.
4. Le noyau lance le processus /init de l'initramfs.
5. Le processus /init vérifie et monte la racine finale spécifiée dans le paramètre root= passé par GRUB à la ligne de commande du noyau.
6. Le processus /init lance le processus /sbin/init ← l'initialisation commence
7. Le processus /sbin/init lance les tâches d'initialisation : montage des systèmes de fichiers définis dans /etc/fstab, activation des consoles virtuelles, démarrage des services...

C'est lors de cette dernière étape que le mode recovery a un effet.

Merci à **al louarn** *et* **raleur** *pour ces précisions sur le post du forum :*

- https://debian-facile.org/viewtopic.php?pid=332625#p332625

Le chargeur de noyau

Le tout premier programme à être lancé est le BIOS. Celui-ci va chercher dans les paramètres du BIOS un chargeur, c'est-à-dire un petit programme destiné à lancer l'OS.

Un chargeur permet aussi à l'utilisateur de lancer un des OS installés dans le même PC.
C'est ce qu'on nomme le **multiboot**.

Les chargeurs les plus utilisés pour lancer un noyau Linux sont *LILO* et *GRUB* et sont capables tous deux de passer des options au noyau.

La configuration pour GRUB :

- Voir le tuto grub2 ici

Le noyau

Au démarrage, le noyau va initialiser le matériel présent à l'aide des drivers des périphériques qu'il contient.

Par contre, les drivers compilés sous forme de modules ne sont pas encore pris en compte.

Sont d'abord initialisés les éléments principaux du PC, comme le processeur, les contrôleurs de périphérique... Les périphériques externes viennent ensuite.
À ce stade, le noyau configure rapidement la carte graphique dans un mode standard pour permettre la lisibilité des opérations par l'utilisateur.

On peut même configurer cela par une option du noyau.

Il y a deux manières d'influer sur ce que va faire le noyau.

1. On peut le recompiler en changeant les paramètres accessibles lors de la configuration ou alors
2. lui passer des options.

Les options peuvent être modifiées d'un démarrage à l'autre de façon temporaire ou définitive. *Cool, non ?* 😎
La recompilation n'est nécessaire que pour des cas particuliers d'ajout de matériels spécifiques par exemple...

Tableau des options

Option	Description
vga=XXX	Modifie la résolution d'écran utilisée pendant le démarrage.
no-scroll	Désactive le défilement du texte sur l'écran.
noapic	L'APIC permet à plusieurs périphériques de partager des ressources communes (les IRQ). Avec cette option, on désactive ce mécanisme en cas de problème avec certains matériels.
mem=XXX	Indique la valeur de la mémoire vive présente sur la machine. Par exemple 512M désignera 512 Méga-octets de mémoire.
init=XXX	Indique le programme lancé après l'initialisation du noyau.

Et d'autres encore...

Le processus init

Une fois que le noyau a fini l'initialisation, il lance le programme `init` qui va devenir le père de tous les autres processus et portera l'identifiant[8] 1.
Pour trouver *init*, le *noyau* va lancer sa recherche dans ces répertoires :

- /sbin/init
- /etc/init
- /bin/init
- /bin/sh

Dès qu'il trouve *init*, le *noyau* lui passe immédiatement le contrôle de la suite des événements.

1. S'il ne le trouve pas, le noyau tentera de lancer un shell pour permettre un accès au système.
2. Si le shell ne fonctionne pas, le noyau indique l'erreur et s'arrête.

Pour modifier la recherche on utilise l'option du noyau **init=** en lui indiquant le chemin complet[9] de l'exécutable à lancer.

Configuration d'init

Le fichier *init* se configure dans le répertoire : `/etc/inittab`.
Exemple de configurations :

```
id:5:initdefault:
```

```
si::sysinit:/etc/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Explication en tableau :

Position	**Nom**	**Description**
1	Identifiant	Une chaîne de caractère choisie par l'utilisateur (sauf dans certains cas particuliers) et permettant d'identifier la ligne.
2	Niveaux d'exécution	Les niveaux d'exécution pour lesquels cette ligne doit être prise en compte.
3	Action	Contient une des actions prédéfinies indiquant ce qui doit être fait.
4	Programme	Le programme qui doit être exécuté lorsque l'on rentre dans les niveaux indiqués.

Certains champs peuvent être ignorés selon ce qui est choisi.

Configuration d'inittab

Tableau de configuration pour `/etc/inittab` (*les configurations principales...*)

Action	**Champs ignorés**	**Description**
initdefault	Programme	Permet d'indiquer le niveau d'exécution à utiliser par défaut. Le champ Niveaux d'exécution contiendra alors une seule valeur qui sera ce niveau par défaut.
sysinit	Niveaux d'exécution	Le champ Programme contient le chemin vers un exécutable qui sera lancé en tout premier par init (donc juste après que le noyau ait terminé ses initialisations).
wait	Aucun	Lorsque le système passera dans la niveau d'exécution spécifié, init exécutera la commande indiquée puis attendra qu'elle se termine.
respawn	Aucun	Semblable à wait si ce n'est qu'à chaque fois que le programme se termine, init le relancera.
ctrlaltdel	Niveaux d'exécution	Permet d'indiquer une commande devant être exécutée lorsque l'utilisateur presse la combinaison de touches Ctrl+Alt+Suppr

Les modes d'exécution

Selon les services qui sont lancés, plusieurs modes d'exécution sont possibles.
La convention appliquée est appelée `System V init`. Elle définit la gestion des différents niveaux.

Dans le fichier **inittab** c'est le programme `/etc/rc.d/rc` qui gère cela. Il est lancé avec un numéro

en paramètre qui définit le niveau d'exécution attribué.

Il y a habituellement 7 niveaux d'exécution numérotés de 0 à 6.

Tableau des niveaux d'exécution

Numéro	Désignation	Description	
0	Arrêt	Ce niveau provoque un arrêt de la machine.	
1	Maintenance	Accès à un shell sans aucun service de lancé. Utilisé pour le dépannage.	
2	Multi-utilisateurs simple	Ouverture à plusieurs utilisateurs en mode texte. Mais les services sont limités (pas de réseau par exemple).	
3	Multi-utilisateurs complet	Ajoute le démarrage de tous les services nécessaires pour plusieurs utilisateurs connectés en mode texte.	
4	Mode utilisateur	Librement utilisé.	
5	Graphique	Identique au mode 3.	Les utilisateurs peuvent se connecter en mode graphique et disposer d'un gestionnaire de fenêtre.
6	Redémarrage	Redémarre la bécane.	

mode initdefault

Après le démarrage, le système se trouve dans le mode indiqué par **initdefault** dans `/etc/inittab`.

Pour le changer, il existe un outil appelé *telinit*. Il suffit de le lancer en lui passant en paramètre un numéro du niveau.
Par exemple :

```
telinit 6
```

Cette commande redémarre la machine.

Le programme /etc/...

En cours de création... smolski *06-05-2013*

Le programme gérant les niveaux d'exécution va consulter le contenu du répertoire `/etc/??/??X.?` (où X correspond au numéro du niveau devant être changé).

1. Il y recherche d'abord les exécutables commençant par la lettre K (pour Kill) suivie de deux chiffres.
2. Il lance ces programmes en leur passant en paramètre *stop*. Cela correspond aux services qui

doivent être arrêté dans ce mode-là.
3. Ils sont lancés dans l'ordre croissant du nombre indiqué après le K.

Ensuite au tour des programmes dont le nom commence par la lettre S (pour Start) avec, pareillement, le nombre sur deux chiffres.
Ils sont lancés de la même manière que les précédents, mais avec le paramètre *start*.

Ces fichiers sont des scripts shell et le même sera lancé pour le démarrage ou l'arrêt.
C'est donc de la responsabilité du script de voir s'il a été appelé avec le paramètre start ou stop pour savoir quelle action entreprendre.

Les distributions GNU/Linux incluent ce genre de script lorsqu'un service est installé. L'utilisateur n'a donc ensuite qu'à modifier les liens symboliques.

Voici un exemple ... (*à suivre...*)

La connexion en mode texte

Après que le système ait terminé les actions à entreprendre dans un niveau donné, un utilisateur peut alors se connecter au système.
Lorsque cela se fait en mode texte, un message invite la personne à entrer son nom d'utilisateur puis son mot de passe.

C'est généralement le programme **login** qui se charge de ça. Une fois le nom et le mot de passe saisis, ce programme va lire le fichier `/etc/passwd`[10] pour vérifier ces informations.

En mode texte, le programme permettant de se connecter va donc lancer le shell choisi par l'utilisateur. Sur un système GNU/Linux, c'est par défaut : bash.

Bash (comme les autres shells) va alors lancer un script dans le répertoire de l'utilisateur. Il s'agit dans ce cas de `.bashrc` qui est un fichier caché.
C'est donc dans ce script qu'un utilisateur peut rajouter des commandes qu'il souhaite voir exécutées lorsqu'il se connecte.

La connexion en mode graphique

Lors d'une connexion en mode graphique, l'utilisateur doit saisir son nom d'utilisateur et son mot de passe dans une fenêtre créée par ce que l'on appelle souvent un gestionnaire d'affichage.
Il existe plusieurs de ces programmes. Les plus connus sont **KDM** et **GDM**.
Ils offrent des fonctionnalités équivalentes : Afficher la liste des utilisateurs, vérifier le mot de passe...
Et surtout lancer ensuite le gestionnaire de fenêtre/bureau choisi par l'utilisateur.

L'environnement graphique sélectionné va ensuite se lancer. Cela correspondra à lancer le gestionnaire de fenêtre, éventuellement un gestionnaire de bureau, puis d'autres applications.
Une fois que ceci est terminé, il y a la possibilité pour l'utilisateur d'avoir des programmes lancés automatiquement.

Selon les cas, choisir ces programmes peut varier fortement. Toutefois, il existe une organisation,

`freedesktop.org`, qui tente d'uniformiser tout ce qui touche aux environnements graphiques.

Très souvent, un répertoire sera donc utilisé pour stocker les applications devant être lancées (dans la pratique, il contiendra des liens symboliques). Ce répertoire est `Desktop/Autostart/` qui se trouve dans le répertoire personnel de l'utilisateur.
Il suffira donc d'y ajouter ce que l'on souhaite.

A noter également que la plupart des gestionnaires de bureau récents incluent aussi un gestionnaire de session.
Le rôle de ce programme est de sauvegarder l'état dans lequel est l'environnement de l'utilisateur quand il se déconnecte (principalement quelles sont les applications lancées).
Il va ensuite le restaurer lorsque le même utilisateur se reconnecte sur le système.

Cette restauration comprend notamment le lancement des applications qui étaient lancées lors de la fermeture de session précédente.

Fichiers

Tout est fichier.

Dans Unix, presque tout apparaît ou peut être manipulé comme un fichier :
les périphériques (/dev) comme les disques, consoles et terminaux, les interfaces avec le noyau (/proc, /sys), les sockets de communication réseau ou inter-processus... On peut communiquer avec eux comme on lit ou écrit dans un fichier.

Une exception notable est les interfaces réseau.

Merci aux intervenants du post ayant permis l'apport de cette indication :

- https://debian-facile.org/viewtopic.php?id=25013

Processus

- Les processus dans le détail.

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !
2)
En premier lieu : le BIOS
3)
la mémoire vive
4)
ou **programmes**
5)
Voir sur Wikipédia : Multitâche préemptif
6)
le système d'exploitation
7)
le **driver**
8)
le PID

9)

Voir : Les chemins absolus et relatifs en détail.

10)

passwd