

SYSTÈME D'EXPLOITATION PARTIE 2





La gestion des réseaux

Système d'exploitation : Gestion des réseaux

Gestion des interfaces réseaux sous Linux:

➤ **Netplan** : (syntaxe **basée sur l'indentation**)

La gestion des interfaces réseaux sous Linux peut être effectuée de différentes manières, selon la distribution et l'outil utilisé. Netplan est un outil relativement récent pour la configuration des interfaces réseau, principalement utilisé dans les systèmes Ubuntu à partir de la version 18.04 et plus récentes.

Netplan est un utilitaire de configuration réseau qui permet de configurer les interfaces réseau à travers des fichiers de configuration en YAML (YAML Ain't Markup Language). Il est principalement utilisé sur les systèmes qui reposent sur systemd comme gestionnaire de système. Cela inclut Ubuntu et d'autres distributions qui utilisent systemd pour gérer les processus du système.

Système d'exploitation : Gestion des réseaux

Gestion des interfaces réseaux sous Linux:

➤ Netplan :

La configuration des interfaces réseau avec Netplan repose sur des fichiers de configuration stockés dans le répertoire /etc/netplan/

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      addresses:
        - 192.168.1.10/24
      gateway4: 192.168.1.1
      nameservers:
        addresses:
          - 8.8.8.8
          - 8.8.4.4
```

Système d'exploitation : Gestion des réseaux

Gestion des interfaces réseaux sous Linux:

➤ **Netplan** : (explication)

network : C'est l'élément racine qui contient toute la configuration réseau.

version: 2 : Spécifie la version du format de configuration (la version 2 est la plus récente et utilisée dans Ubuntu 18.04 et au-delà).

renderer: networkd : Définit le moteur utilisé pour gérer la configuration réseau. networkd est utilisé dans les serveurs, tandis que NetworkManager pourrait être utilisé dans un environnement de bureau.

eth0 : C'est le nom de l'interface réseau. Cela peut être **eth0, ens33, enp0s3**, ou tout autre nom d'interface, selon le système.

dhcp4: false : Indique que l'interface ne doit pas utiliser le protocole DHCP. Dans ce cas, l'adresse IP est statique.

addresses : Liste des adresses IP assignées à l'interface. Ici, 192.168.1.10/24 définit une adresse IP statique avec un masque de sous-réseau de 255.255.255.0.

gateway4 : Définit la passerelle par défaut pour les adresses IPv4. Ici, 192.168.1.1 est la passerelle.

nameservers : Spécifie les **serveurs DNS** à utiliser pour la résolution de noms. Dans cet exemple, les serveurs DNS de Google (8.8.8.8 et 8.8.4.4) sont utilisés

Système d'exploitation : Gestion des réseaux

Gestion des interfaces réseaux sous Linux:

➤ Netplan :

Règles d'indentation dans des fichiers YAML (YAML Ain't Markup Language)

Espaces uniquement (pas de tabulations) : YAML ne permet pas l'utilisation de tabulations (tabs) pour l'indentation. Vous devez utiliser uniquement des espaces. Une tabulation dans un fichier YAML provoque une erreur de syntaxe.

Indentation avec des espaces : Les niveaux d'indentation sont créés en utilisant des espaces, et chaque niveau d'indentation est représenté par deux espaces par niveau, bien que vous puissiez utiliser d'autres multiples d'espace, tant que vous êtes cohérent dans tout le fichier.

Hiérarchie des éléments : L'indentation sert à indiquer les relations hiérarchiques entre les éléments. Par exemple :

network est au niveau supérieur.

version, renderer, et ethernets sont des éléments enfants de network.

eth0, dhcp4, addresses, etc., sont des sous-éléments de ethernets.

Système d'exploitation : Gestion des réseaux

Gestion des interfaces réseaux sous Linux:

➤ Netplan :

Avant d'appliquer la configuration, vous pouvez tester la syntaxe du fichier YAML avec un validateur YAML en ligne, ou utiliser la commande **sudo netplan try** pour tester les modifications sans les appliquer définitivement.

Une fois que vous avez modifié ou créé un fichier YAML dans `/etc/netplan/`, vous devez appliquer la configuration. Pour ce faire, utilisez la commande suivante : **netplan apply**

L'intérêt de Netplan réside dans le fait qu'il offre une configuration simplifiée et unifiée des interfaces réseau, avec l'utilisation d'un format YAML lisible et facilement compréhensible.

Système d'exploitation : Gestion des réseaux

Gestion des interfaces réseaux sous Linux:

➤ Les commandes utiles :

Ip a ou **ip address show** : commande pour afficher les interfaces réseaux

Ethtool eth0 : commande pour afficher les détails d'une interface spécifique

Ip link set eth0 up : commande pour activer l'interface réseau (eth0 le nom de l'interface)

Ip link set eth0 down : commande pour désactiver l'interface réseau

Ip addr add 192.168.1.1/24 dev eth0 : commande pour assigner une adresse IP pour une interface spécifique

Ip addr del 192.168.1.1/24 dev eth0 : commande pour supprimer une adresse IP pour une interface spécifique

IP route show : commande pour afficher la table de routage (route -n besoin d'installer net-tools)

Les Scripts



Système d'exploitation : les scripts

Les différents shells sous linux :

Le **Shell** est l'interface de ligne de commande qui permet à un utilisateur de communiquer avec le noyau du système d'exploitation. Sous Linux, il existe plusieurs types de shells, chacun ayant ses propres caractéristiques et fonctionnalités.

Système d'exploitation : les scripts

Bash (Bourne Again Shell) :

Le **Bash** est le shell le plus utilisé sous Linux et est la version améliorée du Bourne Shell (sh). Il est extrêmement puissant et flexible, et permet d'exécuter des commandes, de créer des scripts, et de personnaliser l'environnement de travail.

```
root@ubuntu:/home# echo "Welcome"
Welcome
root@ubuntu:/home#
```

Système d'exploitation : les scripts

Créer un Script Bash sous Linux :

Les scripts Bash sont des fichiers contenant une série de commandes que vous pouvez exécuter en une seule fois. Ils sont utilisés pour automatiser des tâches répétitives, administrer le système, ou exécuter des programmes complexes.



Systeme d'exploitation : les scripts

Comment créer et exécuter un script bash?



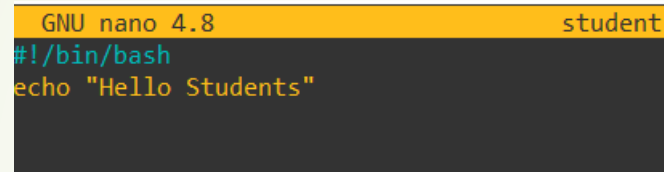
Système d'exploitation : les scripts

Pour créer un script bash:

1. Créer un fichier de script : Vous pouvez créer un script Bash en utilisant un éditeur de texte comme nano, vim, ou vi
2. Écrire le script : Commencez toujours un script avec la ligne suivante pour indiquer à votre système que ce script doit être exécuté avec Bash : **#!/bin/bash**
3. Rendre le script exécutable : Comme discuté durant nos séances de cours, par défaut, un script n'est pas exécutable. Donc vous devez utiliser la commande **chmod** pour lui donner la permission d'exécution.
4. Exécuter le script avec la commande suivante: ./ suivi du nom du script que vous venez de créer, sans mètre d'espace.

Système d'exploitation : les scripts

Voici un exemple de script simple qui affiche "Hello students" lorsqu'il est exécuté :

A screenshot of a terminal window showing the GNU nano 4.8 editor. The editor is open to a file named 'student'. The content of the file is a shell script starting with a shebang line '#!/bin/bash' followed by an 'echo' command that prints 'Hello Students'.

```
GNU nano 4.8 student
#!/bin/bash
echo "Hello Students"
```

La commande **echo** sous Linux et dans les scripts Bash est une commande très courante utilisée pour afficher du texte ou des variables à l'écran, C'est une commande simple, mais très utile pour l'interaction avec l'utilisateur ou pour afficher des informations pendant l'exécution de scripts.

Système d'exploitation : les scripts

Utilisation des variables et arguments dans un script bash:

Trois types de variables sont très courants dans les scripts Bash et permettent de gérer les données d'entrée de manière flexible et dynamique.

1. Variables locales :

Ce sont des **variables définies directement dans le script Bash**. Elles sont utilisées pour stocker des données temporaires pendant l'exécution du script et ne sont valables que dans le contexte de ce script.



```
GNU nano 4.8 student
#!/bin/bash
nom="HEI"
echo "Hello $nom!"
```

Résultat

```
rado@ubuntu:~$ ./student
Hello HEI!
```


le symbole **\$** est utilisé pour accéder à la valeur d'une variable

Système d'exploitation : les scripts

Utilisation des variables et arguments dans un script bash:

2. Variables positionnelles (paramètres passés au script) :

Ces variables sont les arguments passés au script lorsque celui-ci est exécuté. Ces variables sont numérotées et accessibles via \$1, \$2, \$3, etc. Le premier argument passé au script correspond à \$1, le deuxième à \$2, et ainsi de suite.



```
GNU nano 4.8 student
#!/bin/bash
echo "Le premier argument est : $1"
echo "Le deuxième argument est : $2"
```

Résultat

```
rado@ubuntu:~$ ./student K1 K2
Le premier argument est : K1
Le deuxième argument est : K2
```

Système d'exploitation : les scripts

Utilisation des variables et arguments dans un script bash:

3. Variables d'entrée utilisateur :

Ces variables sont obtenues à partir de l'entrée utilisateur via la commande **read**. Le terminal "invite" l'utilisateur à entrer une valeur, qui est ensuite stockée dans une variable que tu as définie.

```
GNU nano 4.8                                student
#!/bin/bash
echo "Vous êtes dans quel groupe ?"
read nom
echo "Bonjour, je suis dans le groupe $nom!"
```

Lorsque ce script est exécuté, le terminal affichera :

```
rado@ubuntu:~$ ./student
Vous êtes dans quel groupe ?
```

Puis l'utilisateur pourra entrer une valeur qui sera stockée dans la variable nom (par exemple K3)

```
K3
Bonjour, je suis dans le groupe K3!
rado@ubuntu:~$
```

Les structures conditionnelles et les boucles dans les scripts



Système d'exploitation : les scripts

Utilisation des conditions et boucles dans un script bash:

Les scripts Bash permettent également l'utilisation de conditions (if), de boucles (for, while), et de tests pour créer des scripts plus complexes.

Système d'exploitation : les scripts

La condition if dans un script bash :

La condition **if** dans un script Bash est utilisée pour exécuter un bloc de code uniquement si une condition est vraie. La syntaxe de base de l'instruction if en Bash est la suivante :

```
if [ condition ]; then  
    # Code à exécuter si la condition est vraie  
else  
    # Code à exécuter si la condition est vraie  
...  
fi
```



Pour les programmes comportant plusieurs conditions, vous pouvez également utiliser la syntaxe "case". N'hésitez pas à faire quelques recherches à ce sujet pour en savoir plus.

Système d'exploitation : les scripts

La boucle for dans un script bash :

La boucle for permet de répéter une série d'instructions un nombre défini de fois, ou pour chaque élément dans une liste ou un ensemble de valeurs.

Les différentes manières d'utiliser une boucle for dans un script bash :

1. Boucle for avec une plage de nombres

La syntaxe de base pour une boucle for avec une plage de nombres est la suivante :

```
for i in {1..5}; do
```

```
    # Code à exécuter si la condition est vraie
```

```
done
```

Ici la boucle va s'exécuter pour i allant de 1 à 5

Système d'exploitation : les scripts

2. Boucle for avec des incréments personnalisés

Une boucle for avec la possibilité possible de spécifier un incrément différent (autre que 1) :

```
for i in {1..10..2}; do  
    # Code à exécuter si la condition est vraie  
done
```

Ici i s'incrémente de 2 à chaque itération

Système d'exploitation : les scripts

3. Boucle for sur une liste d'éléments

Une boucle for avec une liste d'éléments comme itération.

```
for groupe in "K1" "K2" "K3" "K4"  
do  
    echo "Je suis dans le groupe $groupe"  
done
```

Ici la boucle va afficher chaque groupe de la liste un par un.

Système d'exploitation : les scripts

4. Boucle for avec des fichiers dans un répertoire (un peu plus complexe)

Une boucle for avec une itération sur les fichiers d'un répertoire. Par exemple, pour afficher tous les fichiers .txt dans le répertoire courant :

```
for file in *.txt  
do  
    echo "Fichier : $file"  
done
```

Système d'exploitation : les scripts

La boucle while dans un script bash :

La boucle while en Bash est utilisée pour exécuter une série d'instructions tant qu'une condition donnée est vraie. Elle permet d'exécuter les instructions à répétition tant que la condition de la boucle reste vraie.

La syntaxe de base pour une boucle while est la suivante :

```
while [ condition ]; do
```

```
    # Code à exécuter si la condition est vraie
```

```
done
```



Système d'exploitation : les scripts

La boucle while dans un script bash :

Une boucle while peut également être utilisée pour créer une boucle infinie. Par exemple

```
while true  
do  
    echo "Cette boucle s'exécutera indéfiniment."  
    sleep 1  
done
```



sleep 1 : Pause d'une seconde avant de recommencer

Système d'exploitation : les scripts

Liste complète des opérateurs utilisables dans une condition en Bash, classés par type :

1. Opérateurs de comparaison numérique (pour comparer des valeurs numériques):

- eq : égal à
- ne : différent de
- lt : inférieur à
- le : inférieur ou égal à
- gt : supérieur à
- ge : supérieur ou égal à

Système d'exploitation : les scripts

2. Opérateurs de comparaison de chaînes de caractères (pour comparer des chaînes de texte) :

- = : égal à (utilisez avec des guillemets)
- != : différent de
- < : inférieur à (dans l'ordre lexicographique)
- > : supérieur à (dans l'ordre lexicographique)
- z : chaîne vide (vérifie si la chaîne a une longueur de 0)
- n : chaîne non vide (vérifie si la chaîne a une longueur supérieure à 0)

Système d'exploitation : les scripts

3. Tests de fichiers (pour tester des propriétés de fichiers (existence, type, etc.)) :

- e : existe (peu importe le type)
- f : est un fichier régulier
- d : est un répertoire
- r : est lisible
- w : est modifiable (écriture)
- x : est exécutable
- s : est un fichier non vide
- L : est un lien symbolique
- h : est un lien symbolique (même fonction que -L)

Système d'exploitation : les scripts

4. Opérateurs logiques (pour combiner plusieurs conditions avec des opérateurs logiques) :

! : négation (inverse le résultat d'une condition)

&& : ET logique (les deux conditions doivent être vraies)

|| : OU logique (l'une des conditions doit être vraie)

5. Tests de variables spéciales (pour tester les propriétés des variables) :

-z : Vérifie si une chaîne est vide.

-n : Vérifie si une chaîne n'est pas vide.

Système d'exploitation : les scripts

6. Tests d'égalité de variables avec `[[...]]` :

`[[...]]` est utilisé pour des tests plus complexes, tels que la comparaison de chaînes avec des caractères génériques.

`==` : égal à (fonctionne avec les expressions régulières et les jokers)

`!=` : différent de (fonctionne aussi avec des jokers)

`>` : supérieur à (dans l'ordre lexicographique)

`<` : inférieur à (dans l'ordre lexicographique)

Système d'exploitation : les scripts

7. opérateurs de comparaison utilisés pour comparer deux fichiers en fonction de leur date de modification :

- nt : Plus récent que
- ot : Plus ancien que
- ef : Identité des fichiers

Système d'exploitation : Les scripts



Notes importantes :

- **Vous devez ajouter des espaces autour des crochets.**

Exemple incorrect : `[1 -eq 1]` (manque d'espace). Correct : `[1 -eq 1]`.

- **Les crochets simples `[]` (ou Tests conditionnels simples) ne supportent pas les opérateurs logiques complexes `&&`, `||` directement.**

- **Les doubles crochets `[]` sont de syntaxe améliorée par rapport à `[]` :**


Prise en charge des opérateurs logiques `&&`, `||` sans nécessiter d'échappement. Comparaison avancée de chaînes (avec des expressions régulières).



La gestion des processus

Système d'exploitation : les processus

La gestion des processus sous Linux est un aspect essentiel de l'administration du système. Un processus est un programme en cours d'exécution, et Linux propose plusieurs commandes pour gérer, surveiller, et contrôler ces processus.



```
systemd+ 242818 0.0 0.3 26580 6196 ? Ss Jan06 0:02 /lib/systemd/systemd-networkd
root 242956 0.0 0.0 0 0 ? I< Jan06 0:00 [kworker/u257:0-hci0]
root 281860 0.0 0.0 0 0 ? I 18:44 0:02 [kworker/0:1-events]
root 281905 0.2 0.0 0 0 ? I 18:44 0:10 [kworker/1:2-memcg_kmem_cache]
root 281908 0.0 0.0 0 0 ? I 18:44 0:00 [kworker/u256:3-events_power_efficient]
root 282208 0.0 0.0 0 0 ? I 18:53 0:00 [kworker/1:0-events]
root 282697 0.1 0.0 0 0 ? I 19:09 0:04 [kworker/0:2-events]
root 283028 0.0 0.4 13936 8912 ? Ss 19:19 0:00 sshd: rado [priv]
rado 283161 0.0 0.2 14068 5936 ? S 19:19 0:00 sshd: rado@pts/0
rado 283163 0.0 0.2 7196 5296 pts/0 Ss 19:19 0:00 -bash
root 283703 0.0 0.0 0 0 ? I 19:41 0:00 [kworker/u256:0-events_unbound]
root 283837 0.0 0.0 0 0 ? I 19:47 0:00 [kworker/u256:1-events_unbound]
rado 284245 0.0 0.1 7644 3224 pts/0 R+ 20:04 0:00 ps aux
```


Système d'exploitation : les processus

Un processus sous Linux peut être dans différents états, tels que en cours d'exécution, en attente, ou en veille. Chaque processus a un identifiant unique appelé PID (Process ID). Les processus peuvent également avoir un PPID (Parent Process ID), qui indique le processus parent ayant lancé ce processus.

Système d'exploitation : les processus

Commandes principales pour gérer les processus :

1. ps (Process Status)

La commande `ps` affiche les processus en cours d'exécution sur le système (c'est l'équivalent de la gestionnaire des tâches sous windows).

```
radio@ubuntu:~/EXO$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.5 103908 11652 ?        Ss   Jan04   0:31 /sbin/init auto automatic-ubiquity nop
root           2  0.0  0.0      0     0 ?        S    Jan04   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Jan04   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Jan04   0:00 [rcu_par_gp]
```

a : Affiche les processus de tous les utilisateurs.

u : Affiche l'info détaillée sur les processus (comme l'utilisateur, la mémoire utilisée, etc.).

x : Affiche aussi les processus sans terminal associé (par exemple les services).

Système d'exploitation : les processus

USER : Nom de l'utilisateur.

PID : Identifiant du processus (Process ID).

%CPU : Pourcentage d'utilisation du processeur.

%MEM : Pourcentage de la mémoire utilisée par le processus.

VSZ : Taille virtuelle du processus (en Ko).

RSS : Taille de la mémoire physique utilisée (en Ko).

TTY : Terminal associé (si applicable).

STAT : Statut du processus (S pour "sleeping", R pour "running", Z pour "zombie", etc.).

START : Heure de démarrage du processus.

TIME : Temps CPU total consommé par le processus.

COMMAND : Commande ou programme associé au processus.

Système d'exploitation : les processus



REMINDER

Remember, read the manual to know all the arguments of a command

Système d'exploitation : les processus

Commandes principales pour gérer les processus :

2. top

La commande top fournit une vue dynamique en temps réel des processus en cours, triée par l'utilisation du processeur par défaut. Elle montre aussi d'autres informations comme la mémoire, l'uptime du système, et plus encore.

```
top - 03:17:24 up 3 days, 17:48,  2 users,  load average: 0.04, 0.03, 0.00
Tasks: 200 total,  1 running, 199 sleeping,  0 stopped,  0 zombie
%Cpu(s):  0.2 us,  0.2 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  1939.8 total,   741.2 free,   254.2 used,   944.4 buff/cache
MiB Swap:  2048.0 total,  2048.0 free,    0.0 used.  1509.2 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
730	root	20	0	309520	7952	6476	S	0.3	0.4	12:17.87	vmtoolsd
283161	rado	20	0	14068	5936	4460	S	0.3	0.3	0:00.24	sshd
289764	root	20	0	0	0	0	I	0.3	0.0	0:16.91	kworker/1:1-events
294467	rado	20	0	8036	3616	3100	R	0.3	0.2	0:00.03	top
1	root	20	0	103908	11652	8480	S	0.0	0.6	0:32.20	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.17	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.03	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.83	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kblockd
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq

Système d'exploitation : les processus

Commandes principales pour gérer les processus :

3. kill

La commande kill permet d'envoyer des signaux à un processus. Le plus courant est d'envoyer le signal TERM pour demander la terminaison d'un processus. (c'est l'équivalent de fin de tâche sur windows)

4. nice ou renice

Ces commandes permettent de gérer la priorité d'un processus. Chaque processus sous Linux a une priorité, qui détermine l'allocation du CPU. Plus la valeur de nice est basse, plus la priorité est élevée.

Lancer un processus avec une priorité modifiée :

nice -n 10 commande (-n 10 définit une priorité plus faible (valeurs négatives augmentent la priorité).

Système d'exploitation : les processus

Commandes principales pour gérer les processus :

Changer la priorité d'un processus en cours :

`renice -n 5 -p PID`

La commande `renice` permet de modifier la priorité d'un processus en cours d'exécution. L'option `-n` définit le niveau de priorité (ici 5), et `-p` permet de spécifier l'ID du processus (PID).

5. Nohup

La commande `nohup` (no hang up) permet de lancer un processus de manière à ce qu'il continue à s'exécuter même si la session « terminal » est fermée. Elle est souvent utilisée pour exécuter des commandes à long terme ou des tâches qui ne doivent pas être interrompues par la déconnexion.

`nohup commande &`



Lorsque vous ajoutez `&` à la fin d'une commande, cette commande s'exécute en arrière-plan, mais elle peut être arrêtée si vous fermez la session ou la connexion (si la session est fermée, le processus est également tué).

Système d'exploitation : les processus

Commandes principales pour gérer les processus :

6. Affichage d'un arbre de processus avec la commande pstree

La commande `pstree` permet d'afficher un arbre des processus, ce qui montre les relations entre les processus parents et enfants.



La gestion des services

Système d'exploitation : la gestion des services

La gestion des services sur Ubuntu Linux repose principalement sur systemd, qui est le système d'initialisation par défaut utilisé depuis Ubuntu 15.04.

systemd est un système d'initialisation et un gestionnaire de services pour Linux. Il prend en charge le démarrage des services et des processus lors du démarrage de l'ordinateur, gère les services en cours d'exécution, et offre des fonctionnalités avancées comme la gestion des journaux système et le contrôle de l'état des services.

C'est le premier processus lancé lors du démarrage du système, avec le PID 1 (Process ID 1). Il initialise le système et démarre tous les services nécessaires pour que l'ordinateur fonctionne.

systemctl est l'outil en ligne de commande permettant d'interagir avec systemd. Il vous permet de contrôler les services (les démarrer, les arrêter, les redémarrer, les activer au démarrage, etc.).

Système d'exploitation : la gestion des services

La gestion des services sur Ubuntu Linux repose principalement sur systemd, qui est le système d'initialisation par défaut utilisé depuis Ubuntu 15.04.

1. Vérifier l'état d'un service :

Pour vérifier l'état d'un service, utilisez la commande systemctl suivante :

```
systemctl status nom_du_service
```

2. Démarrer un service :

Pour démarrer un service, utilisez la commande systemctl suivante :

```
systemctl start nom_du_service
```

Système d'exploitation : la gestion des services

3. Arrêter un service :

Pour arrêter un service, utilisez la commande systemctl suivante :

```
systemctl stop nom_du_service
```

4. Redémarrer un service :

Pour redémarrer un service, utilisez la commande systemctl suivante :

```
systemctl restart nom_du_service
```

5. Recharger la configuration d'un service (sans redémarrer) :

Certaines modifications de configuration n'ont pas besoin d'un redémarrage complet. Dans ce cas, vous pouvez recharger la configuration du service :

```
systemctl reload nom_du_service
```

Système d'exploitation : la gestion des services

6. Activer un service au démarrage (au boot) :

Pour faire en sorte qu'un service démarre automatiquement au démarrage du système :

```
systemctl enable nom_du_service
```

7. Désactiver un service au démarrage (au boot) :

Pour faire en sorte qu'un service ne démarre pas automatiquement au démarrage du système :

```
systemctl disable nom_du_service
```

Système d'exploitation : la gestion des services

8. Vérifier si un service est activé au démarrage :

Pour vérifier si un service est configuré pour démarrer automatiquement au démarrage du système :

```
systemctl is-enabled nom_du_service
```

9. Lister tous les services en cours d'exécution :

Pour afficher tous les services en cours d'exécution :

```
systemctl list-units --type=service
```

10. Voir les journaux d'un service :

Pour consulter les journaux d'un service spécifique :

```
journalctl -u nom_du_service
```




Cron

Système d'exploitation : cron

cron est un utilitaire Unix/Linux permettant de planifier l'exécution de tâches automatiques à intervalles réguliers. Cela peut être utile pour effectuer des sauvegardes, envoyer des rapports, nettoyer des fichiers, ou effectuer toute autre tâche répétitive sans intervention manuelle.

Les cron jobs (tâches planifiées) sont généralement stockées dans un fichier appelé crontab. Il existe plusieurs manières de gérer les tâches cron :

Crontab de l'utilisateur : chaque utilisateur peut avoir son propre fichier crontab.

Crontab système : utilisé par le système pour des tâches globales.

Système d'exploitation : cron

1. Pour éditer votre crontab

`crontab -e`

2. Lister les tâches cron d'un utilisateur


`crontab -l`

3. Supprimer les tâches cron de l'utilisateur

`crontab -r`

Système d'exploitation : cron

4. Syntaxe d'un Crontab



```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
(END)
```

source image : https://fr-wiki.ikoula.com/fr/Ex%C3%A9cuter_automatiquement_des_scripts_sous_Linux

Système d'exploitation : cron

Chaque champ est séparé par un espace ou une tabulation, et vous pouvez utiliser des caractères spéciaux pour affiner la planification :

- * : Chaque valeur possible pour ce champ.
- , : Liste des valeurs séparées par des virgules.
- : Intervalle de valeurs (par exemple, 1-5 pour lundi à vendredi).
- / : Valeur incrémentielle (par exemple, */5 pour toutes les 5 minutes).
- ? : Remplaçant dans certains cas pour ignorer un champ (principalement dans les systèmes de planification avancée comme Quartz, mais pas dans la version standard de cron).

Exemple :

#Programmation d'un reboot tous les Lundi, Mardi, Jeudi, Samedi a 6h30

30 6 * * 1,2,4,6 root reboot



La gestion des paquets

Système d'exploitation : la gestion des paquets

La gestion des paquets est un aspect clé pour maintenir et installer les logiciels nécessaires. Les outils de gestion de paquets les plus courants sur Ubuntu sont **APT** (Advanced Package Tool) et **dpkg**. **APT** est principalement utilisé pour installer, mettre à jour et gérer des paquets à partir des dépôts en ligne, tandis que **dpkg** est un outil plus bas niveau utilisé pour installer des paquets locaux (fichiers .deb).

Système d'exploitation : la gestion des paquets

1. Mettre à jour les dépôts

Il est important de mettre à jour la liste des dépôts pour s'assurer que vous obtenez la dernière version des paquets.

```
sudo apt update
```

Les **dépôts** (ou **repositories** en anglais) sont des serveurs ou des espaces en ligne où les paquets logiciels sont stockés et accessibles pour installation sur votre système d'exploitation. Dans le cas d'Ubuntu (et d'autres distributions basées sur Debian), les dépôts contiennent des paquets précompilés (fichiers binaires) que vous pouvez installer sur votre système.

Système d'exploitation : la gestion des paquets

2. Mettre à jour les paquets installés

Une fois que la liste des paquets est mise à jour, vous pouvez mettre à jour tous les paquets installés sur votre système qui ont de nouvelles versions disponibles.

```
sudo apt upgrade
```

Cette commande met à jour tous les paquets installés avec les dernières versions disponibles dans les dépôts.

```
sudo apt full-upgrade
```

Cette commande permet de mettre à jour les paquets et de gérer les dépendances de manière plus stricte (cela peut installer ou supprimer certains paquets).

Système d'exploitation : la gestion des paquets

3. Installer un paquet

Pour installer un paquet depuis les dépôts officiels, utilisez la commande :

```
sudo apt install nom_du_paquet
```

Cela va télécharger et installer le paquet ainsi que toutes ses dépendances nécessaires.

4. Rechercher un paquet

Pour rechercher les paquets disponibles , utilisez la commande :

```
apt search nom_du_paquet
```

Cela vous montre une liste de paquets dont le nom ou la description correspond à votre recherche.

Système d'exploitation : la gestion des paquets

5. Supprimer un paquet

Pour supprimer un paquet mais conserver ses fichiers de configuration (utile si vous souhaitez le réinstaller plus tard), utilisez la commande :

```
sudo apt remove nom_du_paquet
```

6. Supprimer un paquet et ses fichiers de configuration

Pour supprimer un paquet ainsi que tous ses fichiers de configuration (en cas de désinstallation complète), utilisez la commande :

```
sudo apt purge nom_du_paquet
```

Système d'exploitation : la gestion des paquets

7. Supprimer les paquets inutiles

Après avoir désinstallé un ou plusieurs paquets, il peut rester des paquets inutilisés (dépendances non requises). Pour nettoyer ces paquets, utilisez la commande :

```
sudo apt autoremove
```

Cela supprime tous les paquets qui étaient installés en tant que dépendances mais qui ne sont plus nécessaires à aucun autre paquet.

8. Mettre à jour un paquet spécifique

Si vous souhaitez mettre à jour un paquet spécifique au lieu de mettre à jour tous les paquets, vous pouvez utiliser apt install avec l'option --only-upgrade.

```
sudo apt install --only-upgrade nom_du_paquet
```

Système d'exploitation : la gestion des paquets

9. Gérer les paquets locaux (.deb) avec dpkg

Si vous avez un fichier de paquet local (par exemple, un fichier .deb téléchargé), vous pouvez l'installer avec la commande dpkg.

```
sudo dpkg -i fichier_paquet.deb
```



Si vous rencontrez des erreurs de dépendances après l'installation avec dpkg, vous pouvez les corriger en exécutant :

```
sudo apt --fix-broken install
```

Système d'exploitation : la gestion des paquets

10. Vérifier l'état des paquets

Pour vérifier l'état d'un paquet (si il est installé, si une mise à jour est disponible, etc.), utilisez la commande :

```
apt list --installed
```




**Cette section couvre les
commandes pour la gestion
quotidienne du système.**

Système d'exploitation : gestion quotidienne

Pour traiter et analyser des données dans des fichiers texte ou les résultats d'une commande :

grep : C'est un utilitaire très populaire dans Unix/Linux utilisé pour rechercher des motifs dans un texte ou un fichier. Il permet de filtrer les lignes d'un fichier ou d'un flux qui contiennent un motif donné, et d'afficher ces lignes.

Recherche simple dans un fichier

```
grep "erreur" fichier.log
```

Afficher les lignes avec les numéros de ligne

```
grep -n "erreur" fichier.log
```

Recherche récursive dans un répertoire (Cela recherche le mot "erreur" dans tous les fichiers du répertoire /var/log/ et ses sous-répertoires.)

```
grep -r "erreur" /var/log/
```

Système d'exploitation : gestion quotidienne

awk : C'est un langage de traitement de texte et un outil très puissant qui permet de manipuler des fichiers texte, de traiter des colonnes spécifiques dans un fichier ou une sortie, et de réaliser des opérations mathématiques, de comparaison, etc.

Cas d'utilisation 1 :

```
grep "K2" fichier.txt | awk '{print $1, $2}'
```

Explication :

grep "K2" fichier.txt : Recherchera toutes les lignes de fichier.txt qui contiennent le motif "K2".

| **(pipe)** : Passe les lignes filtrées par grep à la commande suivante.

(et Oui, | permet de combiner deux commandes dans un terminal Linux.)

awk '{print \$1, \$2}' : Affiche la première et la deuxième colonne des lignes filtrées par grep.

Système d'exploitation : gestion quotidienne

Cas d'utilisation 2 :

```
grep "K3" fichier.txt | awk '{sum += $4} END {print sum}'
```

Explication :

grep "K3" fichier.txt : Recherchera toutes les lignes de fichier.txt qui contiennent le motif "K3".

awk '{sum += \$4} END {print sum}' : awk additionne les valeurs de la 4e colonne pour chaque ligne filtrée et affiche la somme totale à la fin.

Système d'exploitation : gestion quotidienne

Pour afficher des informations sur la mémoire RAM plusieurs commandes peuvent être utilisées. Voici les plus courantes :

free :

La commande free est utilisée pour afficher un résumé de l'utilisation de la mémoire sur le système.

free -h

Explication des résultats de la commande free -h

total : mémoire totale disponible.

used : mémoire utilisée.

free : mémoire libre.

shared : mémoire partagée entre les processus.

buff/cache : mémoire utilisée pour le cache et les buffers.

available : mémoire disponible pour les processus sans utiliser la swap.

Système d'exploitation : gestion quotidienne

Pour afficher des informations sur la mémoire RAM plusieurs commandes peuvent être utilisées. Voici les plus courantes :

vmstat :

La commande vmstat (Virtual Memory Statistics) permet d'afficher des informations sur la mémoire, les processus, la pagination, etc.

Cette commande affiche une sortie détaillée sur la mémoire et d'autres statistiques du système.

vmstat -s

cat /proc/meminfo :

La commande cat peut être utilisée pour afficher le contenu de fichiers système, et /proc/meminfo fournit une vue détaillée de la mémoire du système.

Cette commande affiche des informations détaillées sur l'utilisation de la mémoire dans des valeurs plus brutes, telles que les totaux en kilobytes pour la mémoire, les caches, les buffers, etc.

Système d'exploitation : gestion quotidienne

Pour Vérifier l'espace disque et les partitions, voici les plus courantes :

df -h :

df : Affiche l'espace disque utilisé et disponible sur toutes les partitions montées.

-h : Affiche l'espace de manière lisible par l'homme (par exemple, en Mo ou Go).

lsblk :

La commande lsblk liste tous les périphériques de stockage et leurs partitions, avec des informations supplémentaires (comme la taille, le point de montage, etc.).

fdisk -l :

La commande lsblk vérifie les partitions et le format de disque (type de système de fichiers)

Système d'exploitation : gestion quotidienne

rm : Supprime un fichier ou un répertoire.

rmdir : Supprime un répertoire vide.

mv : Déplace ou renomme des fichiers ou des répertoires.

Exemple : **mv fichier.txt /chemin/**

mv old_name new_name pour renommer un fichier.

cp : Copie un fichier ou un répertoire.

Exemple : **cp fichier.txt /chemin/**

ls : Liste les fichiers et répertoires dans un répertoire.

pwd : Affiche le chemin complet du répertoire courant (Print Working Directory).

whoami : Affiche le nom de l'utilisateur actuel (qui est connecté).

uname : Affiche des informations sur le noyau et le système.

history : Affiche l'historique des commandes précédemment exécutées.

Système d'exploitation : gestion quotidienne

cat : Affiche le contenu d'un fichier. Exemple : `cat fichier.txt` affiche le contenu du fichier.

tac : Affiche un fichier ligne par ligne, mais dans l'ordre inverse (par rapport à `cat`).

tail : Affiche les dernières lignes d'un fichier ou le contenu en temps réel.

more : Permet de visualiser un fichier page par page.

less : Semblable à `more`, mais plus flexible et permet de naviguer facilement dans un fichier.

nano : Éditeur de texte en ligne de commande

vi / vim : Éditeurs de texte en ligne de commande avec des fonctionnalités avancées

find : Recherche des fichiers et des répertoires dans un répertoire en fonction de critères spécifiés.