Administration Linux

Surveillance du Système

M1204 - Services sur Réseaux

Patrice Gommery – Octobre 2020

CONSIGNES GENERALES :

Dans tout l'exercice,
remplacez **VMID** par votre numéro de machine

Le **serveur FTP** de la salle se trouve à l'adresse : **172.16.100.1**,
Sur ce serveur, vous disposez d'un compte **mmis1** avec le mot de passe : **PASSWORD**

PRE-REQUIS

Avant d'aborder concrètement la gestion des processus, et pour les besoins de ce TD, nous allons créer un utilisateur **test** qui nous permettra de générer un peu d'activité sur notre serveur :

1. Créez un utilisateur nommé **test** (*mot de passe 123)*
2. Lancer **tmux** et créer "spliter" votre terminal pour avoir deux sessions.
3. Dans une des sessions , loguer vous avec le compte utilisateur **test**
4. Téléchargez le script **test.sh** disponible sur le serveur FTP de la salle (dossier **mmis1/scripts**)
5. Rendez le script exécutable avec la commande **`chmod 750 test.sh`**
6. Lancez le script **`./test.sh 1`** et laissez-le s'exécuter
7. Revenez dans la session **root** pour continuer .

PARTIE 1 : Qui fait quoi ?

1a) La commande w

Non, il n'y a pas d'erreur de frappe, la commande se nomme bien "**w**" et vous affiche la liste des utilisateurs authentifiés sur votre machine et quels processus ils sont en train d'exécuter .

Exécutez la commande w , le résultat obtenu ressemble à ceci :
(bien entendu, les valeurs ne seront pas les mêmes pour vous)

```
 12:25:29 up 8 days, 19:17,  5 users,  load average: 0,69, 0,28, 0,22
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    172.16.205.3     11:56    7.00s  0.03s  0.00s tmux
root     pts/1    tmux(14840).%0   12:07    0.00s  0.02s  0.00s w
root     pts/2    tmux(14840).%1   12:07    7.00s  1.29s  1.22s /bin/bash ./test.sh 1
test     pts/2    -                12:07    7.00s  1.29s  1.22s /bin/bash ./test.sh 1
```

Quelques explications sur le résultat obtenu :
Première ligne :

12:25:29 est tout simplement l'heure de votre système
up 8 days, 19:17 indique la durée de fonctionnement de votre système
(*depuis son dernier lancement ou reboot*)
5 users , le nombre d'utilisateurs actifs (*y compris le système lui-même*) .
Dans notre cas le résultat est faussé , tmux étant compté comme un utilisateur différent.
En réalité, nous n'avons bien que nos 2 utilisateurs : **root** et **test**
load average: 0,69 0,28 0,22 indique La charge en terme du processeur de votre machine. Tant que les valeurs ne dépassent 1 tout va bien. Pour plus d'information, je vous invite à aller voir cette rapide explication : *https://www.linuxtricks.fr/wiki/performance-cpu-avec-vmstat-et-uptime*

Ensuite, nous voyons :
Les utilisateurs connectés (**USER**)
La machine à partir de laquelle les utilisateurs ont ouvert leur session (**FROM**)
L'heure à laquelle ils se sont connectés. (**LOGIN@**)
Depuis combien de temps l'utilisateur n'a pas exécuté de commandes (**IDLE**)
La commande qu'il est en train d'exécuter (**WHAT**)
Dans l'exemple nous voyons que l'utilisateur **root** est en train d'exécuter **tmux** et qu'il est connecté à partir de la machine 172.16.205.3 (*Une machine de la salle H205*) . On voit aussi que l'utilisateur **test** exécute le script "**test.sh**" (*dans un affichage tmux*)

Cette commande est donc une première méthode rapide pour déterminer qui utilise la machine et ce que chacun exécute en termes de programmes . Inconvénient : les informations sont assez limitées, on ne voit ici que les processus lancés par les utilisateurs, pas ceux du système et des services installés.

1b) Les commandes who , uptime

D'autres commandes permettent de voir un résumé de l'activité de vos utilisateurs.
Essayez-les et observez. Le résultat est le même que pour **w**, mais plus ciblé.

PARTIE 2 : La vie des Processus

Les utilisateurs ne sont pas les seuls à exécuter des processus sur votre machine. Le système et les services installés sont autant de programmes qui tournent sur votre machine et donc utilisent le processeur et la mémoire. Voyons quelques commandes pour visionner l'activité de votre machine.

2a) La commande ps

Comme une démonstration vaut mieux qu'un long discours, saisissez la commande **`ps –ef`** et observez le résultat. Vous devriez voir quelque chose qui ressemble à ça :

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         83     2  0 oct.13 ?       00:00:00 [bioset]
root         85     2  0 oct.13 ?       00:00:00 [scsi_eh_0]
root         86     2  0 oct.13 ?       00:00:00 [scsi_tmf_0]
root         87     2  0 oct.13 ?       00:00:00 [scsi_eh_1]
root         88     2  0 oct.13 ?       00:00:00 [scsi_tmf_1]
root         90     2  0 oct.13 ?       00:00:00 [bioset]
root         91     2  0 oct.13 ?       00:00:00 [bioset]
root         97     2  0 oct.13 ?       00:00:02 [kworker/0:1H]
root        127     2  0 oct.13 ?       00:00:00 [kworker/u3:0]
root        139     2  0 oct.13 ?       00:00:02 [jbd2/sda1-8]
root        140     2  0 oct.13 ?       00:00:00 [ext4-rsv-conver]
root        164     1  0 oct.13 ?       00:00:06 /lib/systemd/systemd-journald
root        172     2  0 oct.13 ?       00:00:00 [kauditd]
root        186     2  0 oct.13 ?       00:00:00 [rpciod]
root        187     2  0 oct.13 ?       00:00:00 [xprtiod]
root        197     1  0 oct.13 ?       00:00:00 /lib/systemd/systemd-udevd
root        225     1  0 oct.13 ?       00:00:00 /sbin/rpcbind -f -w
systemd+    227     1  0 oct.13 ?       00:00:03 /lib/systemd/systemd-timesyncd
root        245     2  0 oct.13 ?       00:00:00 [ttm_swap]
root        303     1  0 oct.13 ?       00:00:01 /lib/systemd/systemd-logind
…
Debian-+    671     1  0 oct.13 ?       00:00:00 /usr/sbin/exim4 -bd -q30m
proftpd    3511     1  0 oct.14 ?       00:00:02 proftpd: (accepting connections)
root      14729   353  8 11:56 ?        00:01:52 sshd: root@pts/0
root      14731     1  0 11:56 ?        00:00:00 /lib/systemd/systemd --user
root      14732 14731  0 11:56 ?        00:00:00 (sd-pam)
root      14738 14729  0 11:56 pts/0    00:00:00 -bash
root      14806     2  2 11:58 ?        00:00:29 [kworker/u2:2]
root      14822     2  0 12:04 ?        00:00:00 [kworker/0:2]
root      14834     2  3 12:07 ?        00:00:25 [kworker/u2:1]
root      14838 14738  0 12:07 pts/0    00:00:00 tmux
root      14840     1 42 12:07 ?        00:04:31 tmux
root      14841 14840  0 12:07 pts/1    00:00:00 -bash
root      14848 14840  0 12:07 pts/2    00:00:00 -bash
root      14855 14848  0 12:07 pts/2    00:00:00 login
test      14860 14855  0 12:07 pts/2    00:00:00 -bash
test      14869 14860 32 12:07 pts/2    00:03:27 /bin/bash ./test.sh 1
root      14872     2  0 12:09 ?        00:00:00 [kworker/0:0]
root      14887 14841  0 12:13 pts/1    00:00:00 tload
root      14888     2  0 12:14 ?        00:00:00 [kworker/0:1]
root      14900 14840  0 12:17 pts/3    00:00:00 -bash
root      14908 14900  0 12:18 pts/3    00:00:00 ps -ef
```

Oups !!, beaucoup de trop de lignes ? Normal, vous voyez ici TOUS les processus lancés par TOUS les utilisateurs, ainsi que TOUS les processus lancés au démarrage du système et par chacun des services en exécution sur celui-ci. Comment faire pour filtrer cette liste et ne voir que les services qui nous intéressent ? Une solution simple : le filtre **grep**

Pour ne voir que les processus lancés par l'utilisateur **test** : **`ps –ef | grep test`**

```
test     14860 14855  0 12:07 pts/2    00:00:00 –bash
test     14869 14860 33 12:07 pts/2    00:03:57 /bin/bash ./test.sh 1
root     14919 14900  0 12:19 pts/3    00:00:00 grep test
```

Pour ne voir que le processus du service **proftpd** : **`ps –ef | grep proftpd`**

```
root     14922 14900  0 12:20 pts/3    00:00:00 grep proftpd
```

Que nous permettent de savoir ces informations . Pour chaque ligne :

Le propriétaire (**UID**) du processus : le compte qui a servi à lancer la commande ou le service
le **PID** : L'identifiant du Processus
le **PPID** : L'identifiant du Processus Parent (Essayer **`pstree –p`** pour voir l'arborescence)
L'heure et/ou la date de lancement du processus (**STIME**)
le temps (**TIME**) d'exécution du processus depuis son lancement.
la commande ou le service exécuté (**CMD**) , donc le processus lui-même.

Nous savons donc maintenant quels sont les processus exécutés sur notre machine, mais il serait encore plus intéressant de savoir quels sont les processus les plus actifs et donc les plus consommateurs de ressources machines.

2a) La commande top

Cette commande va nous permettre d'afficher l'activité de notre machine en temps réel et donc de savoir quels sont les processus les plus actifs.

Saisissez la commande **`top`** : (**q** *pour revenir à l'invite de commandes)*

```
top – 12:23:03 up 3 days, 18:44,  7 users,  load average: 2,09, 2,04, 1,67
Tasks:  85 total,   3 running,  82 sleeping,   0 stopped,   0 zombie
%Cpu(s): 70,6 us, 28,7 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,7 si,  0,0 st
KiB Mem :  1020264 total,   671196 free,    52764 used,   296304 buff/cache
KiB Swap:   477180 total,   477180 free,        0 used.   818076 avail Mem


14840 root      20   0   28480   3612   2632 R 55,3  0,4   7:14.61 tmux: server
14869 test      20   0   11180   2908   2688 R 34,8  0,3   5:06.69 test.sh
14806 root      20   0       0      0      0 S  8,9  0,0   0:35.69 kworker/u2:2
  139 root      20   0       0      0      0 S  0,3  0,0   0:02.79 jbd2/sda1–8
    1 root      20   0   57000   6764   5228 S  0,0  0,7   0:04.50 systemd
    2 root      20   0       0      0      0 S  0,0  0,0   0:00.04 kthreadd
    3 root      20   0       0      0      0 S  0,0  0,0   0:00.80 ksoftirqd/0
    5 root       0 –20       0      0      0 S  0,0  0,0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0,0  0,0   0:01.12 rcu_sched
   15 root      20   0       0      0      0 S  0,0  0,0   0:00.09 khungtaskd
   16 root      20   0       0      0      0 S  0,0  0,0   0:00.00 oom_reaper
   17 root       0 –20       0      0      0 S  0,0  0,0   0:00.00 writeback
   18 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kcompactd0
…
```

la même chose juste pour l'utilisateur test : **`top -u test`**

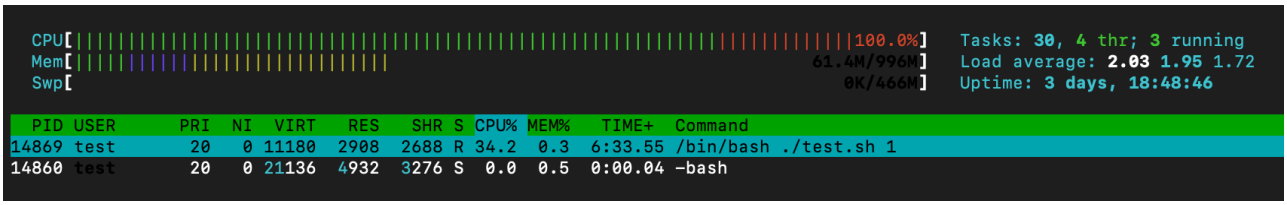
```
top - 12:24:32 up 3 days, 18:45,  7 users,  load average: 1,89, 1,96, 1,67
Tasks:  85 total,   3 running,  82 sleeping,   0 stopped,   0 zombie
%Cpu(s): 77,5 us, 22,5 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1020264 total,   671816 free,    52168 used,   296280 buff/cache
KiB Swap:   477180 total,   477180 free,        0 used.   818692 avail Mem


14869 test      20   0   11180   2908   2688 R 33,8  0,3   5:36.54 test.sh
14860 test      20   0   21136   4932   3276 S  0,0  0,5   0:00.04 bash
```

Nous retrouvons dans les deux cas :
En haut : L'utilisation globale de la machine (Durée, Processeurs, Mémoire, Swap)

Ensuite pour chaque ligne : le **PID**, le Propriétaire du processus (**USER**) mais surtout les % de temps processeur (**%CPU)** et de capacité mémoire (**%MEM**) utilisés en temps réel.

Il devient donc alors facile de détecter un processus qui tournerait en boucle comme le fait ici notre processus (**COMMAND**) **test.sh** lancé par l'utilisateur **test**. On voit ici qu'il consomme à lui tout seul **33,8%** du temps processeur et qu'il tourne depuis **5mn et 36s** .

INFORMATION : Il existe une commande **htop** qui donne une vue plus colorée de la liste des processus. Avant de l'utiliser , il faut d'abord installer le paquet correspond. **Installez-le et testez** .



Exemple avec htop

2c) La commande kill

A quoi peut bien servir de connaître tout cela ? Les exemples d'utilisation de ces informations sont nombreux. Je n'en donnerai ici qu'un seul :

Votre machine est ralentie anormalement. Un petit d'œil à l'activité et vous vous apercevrez que sur votre machine, il existe un processus (*une boucle infinie en php par exemple*) qui consomme à lui tout seul plus de 80% du processeur, ou qu'un autre (*des requêtes SQL mal formatées*) utilise plus de 90% de la mémoire disponible. Que faire ? Rebooter la machine ? Il est clair que ce n'est pas la solution. Et si on arrêtait le processus défaillant .

La commande **kill** est là pour ça . Il suffit juste de connaître le **PID** du processus à arrêter. (vous le voyez avec ps ou top)
Deux méthodes sont disponibles .
La méthode "gentille" : **`kill PID`** (avec PID du processus à arrêter)
La méthode "radicale" : **`kill -9 PID`** (avec PID du processus à arrêter)

Dans le second cas, le processus est tout simplement "éliminé" sans aucun avertissement de la part du système et n'aura donc aucune chance de se terminer proprement. La solution peut paraître violente, mais c'est parfois la seule solution pour débloquer un système.

EXERCICE : Arrêter le script **test.sh** et le service **proftpd** à l'aide de la commande **`kill`**
Vérifiez ensuite avec **`ps -e`** qu'ils sont bien arrêtés.
*Remarquez que l'utilisateur test à toujours un processus actif : **bash** qui correspond au shell et son invite de commande (la session test est toujours ouverte)*

PARTIE 3 : Et si on prenait rendez-vous ?

Vous avez vu en exécutant test.sh, comment lancer un script. Mais pour exécuter ce script, vous avez forcément besoin de vous connecter à la machine pour le lancer. De plus certaines tâches , comme une sauvegarde de données par exemple , ne peuvent pas être effectuées pendant que d'autres utilisateurs sont connectés . Elles peuvent nécessiter d'être exécutées pendant la nuit ou à des heures bien précises. Cette Partie va essayer de répondre à cette problématique.

Pour la suite de ce TP nous désignerons sous le terme de processus, toute commande, service ou script qui s'exécute sur la machine et donc utilise les ressources matérielles de celle-ci.

Dans cette partie nous allons donc traiter trois aspects :

1) Lancer un processus en arrière-plan (ou tâche de fond)
2) Programmer l'exécution d'un processus à une heure donnée
3) Programmer l'exécution d'un processus de façon régulière

3a) Gestion des processus en tâche de fond : "&", `jobs`, `bg`, `fg` et `nohup`

Lorsque vous lancez un script, le temps d'exécution de celui-ci peut être variable en fonction des traitements qu'il exécute. Il peut aussi bien se terminer en quelques secondes qu'en quelques heures. Dans le premier cas, pas de souci, vous récupérez la main (*l'invite de commandes*) presque immédiatement, dans le second, il faut patienter avant de pouvoir saisir une autre commande. Il est même impossible dans le second cas, de fermer le terminal et de se déconnecter, cela mettrait immédiatement un terme au processus (avec un kill violent).

Reprenons l'exemple de notre script **test.sh** . Imaginons que nous voulions le lancer à partir de notre session **root**. Comment faire pour l'exécuter et récupérer l'invite de commande sans l'arrêter, voir même l'exécuter plusieurs fois en simultané ?

Une première solution serait , comme nous l'avons fait jusqu'ici d'ouvrir une nouvelle session dans le terminal , de se connecter de nouveau avec notre utilisateur et de lancer le script. Si nous voulons exécuter le script 10 fois en simultané, il faut faire 10 fois l'opération. Pas génial.

Ce n'est pas parce que nous n'avons pas d'interface graphique et donc que nous ne pouvons pas afficher plusieurs fenêtres dans notre terminal que la chose est impossible. On peut tout à fait lancer plusieurs commandes en simultané à l'intérieur d'un même terminal. (même sans tmux)

IMPORTANT : Avant de continuer, fermez la session de l'utilisateur **test** avec la commande **logout** (ou exit) . La suite du TD est à exécuter dans une session **root**

La première technique consiste simplement à ajouter le signe **&** à la fin de notre commande.
Essayez la commande : **`/home/test/test.sh 2 &`**

Le système vous répond : **`[1] 13197`** (*bien entendu les valeurs sont différentes pour vous*)
A quoi correspondent ces valeurs :
`[1]` est le numéro du processus en arrière-plan
`13197` est le **PID** du processus
Après avoir affiché les informations, le système vous a immédiatement rendu la main en vous affichant de nouveau l'invite de commandes. Vous pouvez donc continuer à travailler.

Attention tout de même. Le processus s'exécute bien en tâche de fond, mais si le processus est censé afficher un résultat à l'écran, il vous l'affichera même si vous êtes en train de saisir une autre commande ou d'afficher un fichier.

Attendez une minute , vous devriez voir s'afficher un message envoyé par le script test.sh

Ceci peut parfois être un peu gênant. Dans ce cas , il est préférable de lancer le processus en redirigeant l'affichage vers un fichier en utilisant les symboles de redirection **>** ou **>>**.

Ce qui donnerait dans notre cas : **`/home/test/test.sh 2 > /root/fichier1.txt &`**
Pour voir le résultat du script, il suffira alors de vérifier le contenu du fichier.

Maintenant, notre script s'exécute en arrière-plan .
Comment savoir s'il est terminé ou pas ? Une commande permet de gérer tout ça: **`jobs`**
Cette commande vous retourne les informations sur les processus d'arrière-plan :

```
[2]+  En cours d'exécution    /home/test/test.sh 2 > /root/fichier1.txt &
```

Dans l'exemple ci-dessus, on voit que le job [2] est toujours en cours d'exécution. Le job [1] est donc terminé. Mais si vous patientez encore une minute, vous devriez voir le message envoyé dans le fichier **/root/fichier1.txt**

Il est aussi possible de passer un processus en arrière-plan en le stoppant avec la combinaison de touches [**CTRL]+Z** . Dans ce cas le processus est stoppé, on peut alors le relancer en arrière-plan avec la commande **bg** suivi de son numéro dans liste des jobs. Essayez :

`/home/test/test.sh 2 > /root/fichier2.txt` *(sans mettre de signe &)*
[CTRL]+Z
`jobs` (pour récupérer son numéro de job : **JID**)
bg JID (**JID** étant son numéro de job)
`jobs` (pour vérifier qu'il est revenu en exécution)

INFORMATION : Vous pouvez aussi repasser un job au premier plan avec la commande **fg** suivi du numéro du job.

Le travail en arrière-plan c'est bien, mais que se passe-t-il si je quitte la session ou tout simplement que je ferme le terminal ? C'est simple tous les processus en cours dans la session , donc y compris exécutés en arrière-plan sont "tués". La seule solution pour éviter ce désagrément est d'utiliser la commande **nohup** . Elle s'utilise comme ceci **nohup** *commande*, elle peut bien sûr être combinée avec le symbole **&** en fin de ligne. Essayez la commande suivante :

```
nohup /home/test/test.sh 2 > /root/fichier3.txt &
```

Le système vous répond :
nohup: entrée ignorée et sortie d'erreur standard redirigée vers la sortie standard

Fermez la session en cours (exit ou logout) et reconnectez-vous .
Exécutez la commande : **ps –e** , vous devriez voir un processus nommé **test.sh** qui correspond au script que vous avez lancé à partir de la session fermée.

Attention tout de même aux commandes exécutées avec **nohup**, il est en effet impossible de repasser le processus au premier plan et donc d'avoir un retour écran d'un éventuel résultat. Dans notre cas, il faudra attendre la fin du script (10mn) avant de pouvoir aller vérifier l'existence du **fichier3.txt** dans notre dossier **/root**

3b) Programmation du lancement d'un processus. La commande at

Il est aussi possible de programmer le lancement d'un processus à une date et une heure donnée, mais il vous faut peut-être avant, installer le paquet **at** qui fournira la commande du même nom.
La commande **at** s'utilise en deux temps :

1) on appelle la commande **at** en lui indiquant (la date) et l'heure désirée pour le lancement
2) à l'invite de commande **at>** , on saisit la commande à exécuter
3) on termine par **[CTRL]+D**

Essayons avec notre script et le test 3, celui-ci va créer un fichier nommé **fichier4.txt** dans notre dossier **/root.** Nous allons programmer l'exécution du script pour dans 5m :

saisissez la commande **at now +5 minutes**
à l'invite de commande **at>** , saisissez le lancement de notre script **/home/test/test.sh 3**
faites **[CTRL]+D** pour fermer la programmation de **at**
Vous pouvez afficher la liste des jobs programmés avec la commande : **atq**

INFORMATION : Vous pouvez aussi supprimer un job programmé avec la commande **atrm** suivi du numéro du jobs.

En attendant que le processus s'exécute, quelques explications sur la façon d'indiquer la date et l'heure derrière la commande **at** . Les options sont les suivantes :

- Une **heure** au format **HH:mm**
- Une **heure** et une **date** au format **MM/JJ/YY** .
  Exemple **13:00 12/01/18** pour le **1er Décembre 2018 à 13h00**
- **now** (maintenant) **+X minutes, hours, days, weeks, months ou years**

Si les 5mn sont passées, vous devriez voir un **fichier4.txt** dans votre dossier **/root**
`cat /root/fichier4.txt` pour vérifier.
Sinon, patientez encore un peu
ou vérifiez avec **atq** que l'exécution du script est toujours en file d'attente.

3c) Pour exécuter un processus régulièrement : crontab

Pour finir cette partie, il ne reste plus qu'à voir comment nous pouvons programmer de façon régulière le lancement d'un processus. Pour cela nous allons utiliser **crontab** .

Cette commande permet de modifier une liste de tâches appelée la "**crontab**" qui contiendra la programmation des processus. Cette liste sera ensuite relue par le programme **cron** qui lui exécutera les processus aux dates et heures programmées.

La commande **`crontab`** s'utilise avec les options suivantes :

-e : Pour éditer la liste **crontab**
-l : Pour afficher la liste
-r : Pour supprimer la liste. Attention TOUTE la liste.

REMARQUE : Il existe une liste "**crontab**" par utilisateur. Les processus seront donc exécutés avec les droits du propriétaire de la liste. Dans notre cas, ils seront donc exécutés avec les permissions root.

Pour la démonstration, nous allons modifier la "crontab" pour exécuter notre script toutes les 5m.
Commencez par éditer la "crontab" avec la commande **`crontab –e`**

REMARQUES :
- La première fois que vous lancez **`crontab –e`** , le système vous demande quel éditeur de texte vous voulez utiliser. Dans notre cas, choisissez **`nano`**.
- Toutes les lignes commençant par # sont des commentaires et ne seront donc pas prises en compte par cron lors de l'exécution des tâches. Il ne faut donc pas commencer votre nouvelle ligne par un #

Ajoutez la ligne suivante à la fin du fichier

```
*/2 * * * * /home/test/test.sh 4
```

Enregistrez les modifications et quittez nano avec [**CTRL]+X** comme d'habitude.
Le système doit vous répondre : **crontab: installing new crontab**
En cas d'erreur, il vous indiquera la ligne à corriger et vous proposera de ré-éditer la liste.

Si vous n'avez pas d'erreur, il ne reste plus qu'à attendre.
En attendant, quelques explications sur la ligne que nous venons d'ajouter dans la "crontab" :

- Chaque ligne commence par 5 valeurs séparées par un espace qui indiquent respectivement :
  - Minutes (0 à 59)
  - Heures (0 à 23)
  - Jour du mois (1 à 31)
  - Mois (1 à 12)
  - Jour de la semaine (0 à 6 avec 0 égal au Dimanche)
- La suite de la ligne indique simplement la commande à exécuter.

Voici quelques exemples de notation pour indiquer le moment choisi pour l'exécution :

`*/5 * * * *`	Toutes les 5mn quel que soit l'heure, le jour et le mois
`* 5,12,19 * * *`	A 5h00,12h00 et 19h00 tous les jours, tous les mois
`30 20 1-15 * *`	Tous les 1er au 15 de chaque mois à 20h30
`* */1 * * 1`	Tous les Heures le Lundi de chaque mois

Si plus de 2mn sont passées depuis la mise à jour de la **"crontab"**, vous devriez voir un **fichier5.txt** dans votre dossier **/root.** Ouvrez le avec la commande **`cat`** et regardez combien de fois il a été exécuté.

Le script c'est bien exécuté toutes les 2mn ? Si c'est le cas vider la "crontab" avec l'option **-r** pour éviter que le fichier ne continue à se remplir.

PARTIE 4 : Il reste de la place ?

Nous savons surveiller l'activité processeur, l'occupation de la mémoire, il ne reste plus qu'un élément à contrôler : **l'espace de stockage** ou plus précisément l'occupation des disques durs de la machine. Pour cela , rien de plus simple, une seule commande suffit : **df**

Sans rentrer dans les détails quelques explications sur la gestion des disques et des partitions sous Linux .

Sous Linux, les partitions ne sont pas identifiées avec des lettres (C,D,..) comme sous windows , mais par des "devices" accrochées à l'arborescence (**/dev**) .
Le premier disque est identifié par **sda**, le second par sdb etc .
Notre première partition sur le premier disque est donc : **/dev/sda1**

Si ne ne voulons afficher que les informations de notre unique disque dur : **`df -h /dev/sda1`**

```
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
/dev/sda1          8,8G    1,4G  7,0G  17% /
```

Nous voyons ici, notre disque dur qui fait donc 8,8Go avec un taux d'occupation de 17%, ce qui laisse encore 7Go de disponible. De quoi faire encore beaucoup de TPs réseaux :-)

Tant que le taux d'occupation ne dépasse pas 80%, on peut considérer que tout va bien, mais si le seuil est dépassé, il serait prudent d'en avertir l'administrateur. Comment faire ? Là encore, plusieurs solutions : Utiliser un outil de monitoring comme **monit** (à venir dans un futur TP) ou exécuter un script qui surveillerai régulièrement (avec crontab) le disque et enverrai un mail à l'administrateur. en cas de dépassement du seuil . (Un exercice à faire chez vous ? :-)

PARTIE 5 : Sauvegardez vos données !!!

Pour finir avec cette rapide visite des possibilités d'administration d'un serveur Linux, il ne serait pas raisonnable d'oublier de parler des sauvegardes. En effet quel que soit votre niveau de compétence et la garantie de fonctionnement assurée par votre hébergeur, personne n'est à l'abri d'une erreur ou d'une application défaillante. Il est donc IMPERATIF d'effectuer des sauvegardes des données utilisateurs, mais aussi des fichiers de configuration du système pour ne pas avoir à tout reconfigurer en cas de "Crash".

Mais d'abord, mettons-nous d'accord sur la définition du terme "Sauvegarde" . Il ne s'agit pas simplement de faire une copie dans un coin de son disque , ou encore pire de simplement dupliquer les fichiers en .save ou .backup dans le dossier . Une "Sauvegarde" c'est surtout une copie régulière des données à sécuriser sur un support externe à la machine. Le support externe peut-être un disque dur, une clé usb ou n'importe quel autre support physique , mais aussi un serveur FTP, un espace de stockage sur le Cloud ou un partage NFS.

Comment sauvegarder ? 2 solutions : utiliser des logiciels dédiés ou simplement utiliser les outils du système mis à notre disposition. C'est la seconde solution que nous allons exploiter pour ce TP.

Quels fichiers sauvegarder ? Avant tout , les données utilisateurs qui stockées dans les dossiers de **/home**, mais aussi les fichiers de configuration qui sont dans **/etc .** En fonction de vos applications, il peut y avoir d'autres dossiers et fichiers à sauvegarder, comme par exemple **/var/www** pour les pages de votre serveur web, ou **/var/log** si on veut sauvegarder les fichiers de logs. Cette liste n'est pas exhaustive et il convient de bien connaître son système pour ne rien oublier.
(Si vous suivez assidument les cours réseaux, cela devrait être plus facile ☺)

4a) Les commandes tar et gzip

Pour simplifier notre sauvegarde, nous allons commencer par créer une archive (**.tar**) de nos données , nos fichiers de configuration et nos fichiers de logs . Il sera ensuite plus simple de transférer ce fichier sur un serveur FTP ou un serveur NAS. La commande idéale pour cette opération est la commande **tar** . La syntaxe est assez simple et l'outil très efficace.

Avant de l'utiliser, commencez par créer un dossier **/backups** qui nous permettra de stocker notre archive. Ensuite archivons nos fichiers dedans avec la commande suivante :
`tar –cvf /backups/backup.tar /etc /home /var/log`
(en des espaces à bien respecter, nous sauvegardons bien 3 dossiers différents)

Si tout se passe bien, vous devriez avoir un fichier nommé **backup.tar** dans le dossier **/backups**.
Ce fichier contient une archive des dossiers **/etc , /home , /var/log** et de leur contenu.
La syntaxe de la commande est assez facile à comprendre :

- -**c** pour créer une archive (**-r** pour rajouter dans une archive existante)
- -**v** pour afficher le détail de l'opération (mode verbeux)
- -**f** pour assembler l'archive dans un fichier

viennent ensuite : le nom du fichier d'archive, suivi du noms des dossiers et fichiers à archiver séparés par un espace. Quelques autres options possibles :

- -**tf** pour afficher le contenu d'une archive sans l'extraire
- -**xvf** pour extraire une archive dans le répertoire courant.

Une dernière chose à faire avant de réellement sauvegarder nos données et donc de les copier sur un support externe : Compresser le fichier d'archive pour qu'il soit plus léger à recopier.

Est-ce bien nécessaire ? Regardez la taille de votre fichier **backup.tar** avec la commande **ls -l**
Sur ma machine, sensiblement la même que la vôtre, il fait un peu plus de 20Mo .
Si nous devons les transférer en FTP cela pourrait être un peu long. Nous allons donc compresser le fichier avec la commande : **`gzip /backups/backup.tar`**

Cette commande créait tout simplement un fichier **backup.tar.gz** dans le dossier **/backups.**
Observez la nouvelle taille du fichier . Dans mon cas, à peine 7Mo, presque trois fois moins que le fichier .tar d'origine. Normal, la plupart des fichiers que nous avons archivés sont des fichiers textes et c'est sur ce type de fichiers que la compression est la meilleure, ce ne serait bien sûr pas le cas avec des fichiers images ou même des vidéos.

Voilà, pour réellement terminer notre sauvegarde , il ne restera plus qu'à transférer ce fichier **`backup.tar.gz`** sur un serveur FTP ,et à automatiser le tout avec un **script** et **crontab**. (Peut-être un autre exercice à faire chez vous ?)

Pour terminer ce TD, renommez votre fichier **backup.tar.gz** en MMI.tar.gz (En remplaçant **MMI** par votre identifiant MMI) et "Uploader" le sur le serveur de l'enseignant **(IP : 172.16.13.26, Identifiant : td06, mot de passe 123)**