

Commandes de recherche avancée

- 1- Pour accéder au répertoire voulu ('Bureau', 'Documents', 'Téléchargement', 'Vidéos' et 'Images') utilisation de la commande `"cd + nom_du_répertoire"`
- 2- Création des cinq fichiers textes nommés 'mon_texte.txt' avec la commande `"echo Que la force soit avec toi > mon_texte.txt"`
- 3- Pour localiser les fichiers textes à l'aide du mot 'force' utilisation de la commande `"find . -name 'mon_texte.txt' -exec grep -l 'force' {} +"`

Compression et décompression de fichiers

- 1- Se déplacer dans le répertoire 'Documents' puis utilisation de la commande `"mkdir Plateforme"` pour créer le répertoire 'Plateforme', utilisation de la commande `"mv mon_texte.txt Plateforme"` pour déplacer le texte 'mon_texte.txt' dans le répertoire 'Plateforme'
- 2- Se déplacer dans le répertoire Documents/Plateforme avec la commande `"cd Documents/Plateforme"`, puis on duplique quatre fois le texte avec les commandes `"cp mon_texte.txt mon_texte_2.txt"`, `"cp mon_texte.txt mon_texte_3.txt"`, `"cp mon_texte.txt mon_texte_4.txt"`, `"cp mon_texte.txt mon_texte_5.txt"`
- 3 - On archive le répertoire Plateforme avec la commande `"tar -czvf Documents/Plateforme.tar.gz Documents/Plateforme/"`
Ou avec bzip2 pour une autre méthode de compression `"tar -cjvf Documents/Plateforme.tar.bz2 Documents/Plateforme/"`

Dans cette commande :

- -c : Créer une archive.
- -z : Utiliser gzip pour compresser.
- -v : Afficher les détails de l'opération.
- -f : Spécifier le nom de l'archive.

- 4 - On désarchive le répertoire Plateforme avec la commande `"tar -xzf Documents/Plateforme.tar.gz -C Documents/"`
Ou avec bzip2 pour une autre méthode de compression `"tar -xjvf Documents/Plateforme.tar.bz2 -C Documents/"`

Dans cette commande :

- -x : Extraire les fichiers de l'archive.
- -z : Utiliser gzip pour décompresser.
- -v : Afficher les détails de l'opération.
- -f : Spécifier le nom de l'archive.

Manipulation de texte

Capture d'écran du script_csv.py dans l'éditeur nano

```
GNU nano 7.2 script_csv.py
#!/usr/bin/env python3

import csv

data = [
    ["Jean", "25 ans", "Paris"],
    ["Marie", "30 ans", "Lyon"],
    ["Pierre", "22 ans", "Marseille"],
    ["Sophie", "35 ans", "Toulouse"]
]

with open('personnes.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(data)
```

[Lecture de 15 lignes]

^G Aide ^O Écrire ^W Chercher ^K Couper ^T Exécuter ^C Emplacement
^X Quitter ^R Lire fich. ^\ Remplacer ^U Coller ^J Justifier ^_ Aller ligne

Pour donner les autorisations d'exécution, utilisation de la commande

`“chmod +x nom_du_script”`

Pour exécuter le script et créer le fichier csv utilisation de la commande

`“./nom_du_script”`

Pour extraire les informations, utilisation de la commande

`“awk -F ',' '{print $3}' personnes.csv”`

Gestion du processus

Capture d'écran avant terminaison du processus **ibus-engine-sim** avec la commande **"ps -p PID"**

```
adammessaoudi@Adam:~/Documents/SystemeScriptSecurite$ ps -p 6002
```

PID	TTY	TIME	CMD
6002	?	00:00:00	ibus-engine-sim

Capture d'écran après terminaison du processus avec la commande

"kill PID", pour vérifier l'état du processus : **"ps -p PID"**

```
adammessaoudi@Adam:~/Documents/SystemeScriptSecurite$ kill 6002
adammessaoudi@Adam:~/Documents/SystemeScriptSecurite$ ps -p 6002
```

PID	TTY	TIME	CMD
-----	-----	------	-----

Capture d'écran avant terminaison du processus **Xwayland** avec la commande **"ps -p PID"**

```
adammessaoudi@Adam:~/Documents/SystemeScriptSecurite$ ps -p 5982
```

PID	TTY	TIME	CMD
5982	?	00:00:00	Xwayland

Capture d'écran après terminaison forcé du processus avec la commande

"kill -SIGKILL PID", pour vérifier l'état du processus : **"ps -p PID"**

```
adammessaoudi@Adam:~/Documents/SystemeScriptSecurite$ kill -SIGKILL 5982
adammessaoudi@Adam:~/Documents/SystemeScriptSecurite$ ps -p 5982
```

PID	TTY	TIME	CMD
			—

Comparaison entre Terminaison Normale et Forcée :

Terminaison Normale (kill PID) :

- Envoie un signal de terminaison au processus.
- Le processus a la possibilité de traiter ce signal et de se terminer proprement.
- Les fichiers ouverts peuvent être fermés correctement, les données en cache peuvent être écrites sur le disque, etc.
- C'est la méthode recommandée pour arrêter un processus lorsqu'il répond.
- Terminaison Forcée (kill -9 PID ou kill -SIGKILL PID) :
 - Envoie un signal de terminaison brutale au processus.
 - Le processus est immédiatement stoppé sans avoir la chance de se terminer proprement.
 - Les fichiers ouverts ne sont pas fermés correctement, ce qui peut entraîner une perte de données.

- C'est une méthode à utiliser en dernier recours lorsque le processus ne répond pas à la terminaison normale.

En comparant les deux méthodes, vous verrez que la terminaison normale permet au processus de terminer ses tâches en cours et de libérer les ressources de manière propre. En revanche, la terminaison forcée arrête immédiatement le processus sans lui permettre de terminer ses tâches en cours.

Surveillance des ressources système

Pour surveiller en temps réel les processus en cours d'exécution, l'utilisation du CPU, de la mémoire et d'autres ressources système, utilisation de la commande :

“top” ou **“vmstat”** ou encore **“free”**

```
top - 14:57:53 up 3:44, 1 user, load average: 0,14, 0,15, 0,18
Tâches: 203 total, 1 en cours, 202 en veille, 0 arrêté, 0 zombie
%Cpu(s): 0,5 ut, 0,3 sy, 0,0 ni, 99,2 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 3880,7 total, 2032,4 libr, 1272,4 util, 820,0 tamp/cache
MiB Éch : 975,0 total, 975,0 libr, 0,0 util, 2608,3 dispo Mem
```

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
5556	adammes+	20	0	3826096	287524	133700	S	1,7	7,2	0:23.76	gnome-s+
7282	adammes+	20	0	557940	50976	38996	S	0,7	1,3	0:00.73	gnome-t+
6107	adammes+	39	19	703432	62200	19888	S	0,3	1,6	0:00.61	tracker+
7142	root	20	0	0	0	0	I	0,3	0,0	0:02.47	kworker+
7440	adammes+	20	0	11828	5660	3492	R	0,3	0,1	0:00.03	top
1	root	20	0	168104	12612	9200	S	0,0	0,3	0:02.40	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.04	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par+
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	slub_fl+
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	netns
8	root	0	-20	0	0	0	I	0,0	0,0	0:00.65	kworker+
10	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_perc+
11	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tas+
12	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tas+
13	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_tas+
14	root	20	0	0	0	0	S	0,0	0,0	0:00.33	ksoftir+

Capture d'écran d'une partie du Script **monitor_system.sh** qui surveille les ressources du système d'exploitation et enregistre les informations dans un fichiers CSV

```
GNU nano 7.2 monitor_system.sh
#!/bin/bash

# Fonction pour surveiller l'utilisation du CPU
monitor_cpu() {
    echo "Timestamp,CPU Usage (%)"
    while true; do
        echo "$(date +%Y-%m-%d %H:%M:%S),$(top -bn1 | grep "Cpu(s)" | sed "s/ /,/" | cut -d, -f1,2)"
        sleep 1
    done
}

# Fonction pour surveiller l'utilisation de la mémoire
monitor_memory() {
    echo "Timestamp,Memory Usage (%)"
    while true; do
        echo "$(date +%Y-%m-%d %H:%M:%S),$(free | grep Mem | awk '{print ($3/$2)*100}')"
        sleep 1
    done
}

^G Aide      ^O Écrire   ^W Chercher ^K Couper   ^T Exécuter ^C Emplacement
^X Quitter   ^R Lire fich.^V Remplacer ^U Coller   ^J Justifier ^_ Aller ligne
```

Ce script crée un fichier CSV avec trois colonnes: **“Timestamp”** (**horodatage**), **“CPU Usage (%)”**, **“Memory Usage (%)”** et **“Other Resource Usage”**

Capture d'écran des informations enregistrées par le script **monitor_script.sh** dans un fichiers CSV nommé **system_usage.csv**

```
GNU nano 7.2 system_usage.csv
Timestamp,CPU Usage (%)
Timestamp,Other Resource Usage
Timestamp,Memory Usage (%)
2024-03-25 15:19:37,55,5285
2024-03-25 15:19:37,25%
2024-03-25 15:19:37,100
2024-03-25 15:19:38,55,5283
2024-03-25 15:19:38,25%
2024-03-25 15:19:39,100
2024-03-25 15:19:39,25%
2024-03-25 15:19:39,55,5283

[ Lecture de 11 lignes ]
^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter   ^C Emplacement
^X Quitter   ^R Lire fich.^N Remplacer  ^U Coller    ^J Justifier  ^_ Aller ligne
```

Capture d'écran de l'exécution du script avec la commande ***“./nom_du_script”***

```
adamessaoudi@Adam: ~/Documents/SystemeScriptSecurite$ ./monitor_system.sh
Timestamp,CPU Usage (%)
Timestamp,Other Resource Usage
Timestamp,Memory Usage (%)
2024-03-25 15:21:06,25%
2024-03-25 15:21:06,56,0539
2024-03-25 15:21:06,100
2024-03-25 15:21:07,25%
2024-03-25 15:21:07,56,0535
2024-03-25 15:21:07,100
2024-03-25 15:21:08,25%
2024-03-25 15:21:08,56,0539
2024-03-25 15:21:09,100
2024-03-25 15:21:09,25%
2024-03-25 15:21:09,56,0539
2024-03-25 15:21:10,100
2024-03-25 15:21:10,25%
2024-03-25 15:21:10,56,0539
2024-03-25 15:21:11,100
2024-03-25 15:21:11,25%
2024-03-25 15:21:11,56,0983
```

Scripting avancé

Capture d'écran du script backup_plateforme.sh

```
GNU nano 7.2 backup_plateforme.sh
#!/bin/bash

# Répertoire parent où se trouve le répertoire Plateforme
parent_directory="$HOME/Documents/SystemeScriptSecurite"

# Répertoire source à sauvegarder
source_directory="$parent_directory/Plateforme"

# Répertoire de sauvegarde
backup_directory="$parent_directory/backup_plateforme"

# Nom du fichier d'historique
log_file="$backup_directory/historique_sauvegardes.txt"

# Fonction pour sauvegarder le répertoire et mettre à jour l'historique
function sauvegarder {
    # Créer le répertoire de sauvegarde si n'existe pas
    mkdir -p "$backup_directory"

    # Nom du répertoire de sauvegarde avec timestamp
    backup_folder="$backup_directory/${date +%Y-%m-%d_%H-%M-%S}"

    # Copier le contenu du répertoire source vers le répertoire de sauvegarde
    cp -r "$source_directory" "$backup_folder"

    # Enregistrer opération dans historique
    echo "$(date +%Y-%m-%d %H:%M:%S) : Sauvegarde effectuée dans $backup_folder"
}

# Exécuter la sauvegarde
sauvegarder
```

Capture d'écran des sauvegardes de l'historique

```
adamnessaoudi@Adam:~/Documents/SystemeScriptSecurite/backup_plateforme$ ls
2024-03-25_16-52-13  2024-03-25_16-53-48  historique_sauvegardes.txt
2024-03-25_16-52-54  2024-03-25_16-54-30
```

```
GNU nano 7.2 historique_sauvegardes.txt
2024-03-25 16:52:13 : Sauvegarde effectuée dans /home/adamnessaoudi/Documents/S
2024-03-25 16:52:54 : Sauvegarde effectuée dans /home/adamnessaoudi/Documents/S
2024-03-25 16:53:48 : Sauvegarde effectuée dans /home/adamnessaoudi/Documents/S
2024-03-25 16:54:30 : Sauvegarde effectuée dans /home/adamnessaoudi/Documents/S
2024-03-25 17:02:02 : Sauvegarde effectuée dans /home/adamnessaoudi/Documents/S
2024-03-25 17:14:01 : Sauvegarde effectuée dans /home/adamnessaoudi/Documents/S
2024-03-25 17:15:01 : Sauvegarde effectuée dans /home/adamnessaoudi/Documents/S
2024-03-25 17:16:01 : Sauvegarde effectuée dans /home/adamnessaoudi/Documents/S
```

```
[ Lecture de 8 lignes ]
^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement
^X Quitter   ^R Lire fich ^M Remplacer ^U Coller    ^J Justifier ^_ Aller ligne
```

Pour automatiser et exécuter le script toute les 1 minutes ouvrir **crontab** avec “**crontab -e**” ajout de la ligne de commande *** * * * *** **chemin/vers/le/répertoire/backup_plateforme**

```
GNU nano 7.2 /tmp/crontab.5ux64o/crontab
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * /home/adamessaoudi/Documents/SystemeScriptSecurite/backup_plateforme>
```

```
^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement
^X Quitter   ^R Lire fich.^M Remplacer  ^U Coller    ^J Justifier ^_ Aller liqne
```