Correction Examen : Système d'Exploitation [SYS1]

Partie 2 : Les commandes avancées pour manipuler du texte

Manipulation de fichiers texte :

Dans cette section, nous allons utiliser les commandes grep, sed et awk pour manipuler des textes. Voici un rappel rapide des commandes :

- **grep** cherche un mot ou une phrase dans chaque ligne d'un fichier.
- **awk** cherche des informations dans les colonnes d'un fichier.
- **sed** permet de reformater le résultat d'une commande.

Nous avons un fichier appelé Liste_eleve qui contient la liste des élèves, exportée d'un fichier CSV avec des virgules comme séparateurs. L'objectif est de filtrer les lignes contenant le mot "ANDRIA", remplacer les virgules par des espaces, et afficher seulement le nom et le premier prénom.

1 - Le fichier Liste_eleve contient des données sur les élèves.

```
root@ubuntu:/home/rado/K3# cat Liste_eleve
STD23002,ANDRIAMARO,Andraina
STD24001,ANDRIANARIVOSON,Ny Aina Nathan
STD24004,RANDRIAMIZARA,Henintsoa Sarobidy Jessy
STD24006,RANDRIANIAINA,Diarilala
STD24014,FANAMBIMANALINA,Mboahangy Fitia Fiorenana
STD24017,RAZAKASOA,Tanjon'Ny Avo Nekena
STD24024,RANDRIANASOLO,Ny Herin'ny Finoana
STD24030,KAMIL HOUSSEN,Tiana Ismael
STD24043,RAHARISOLOMANANA,Johary Niaina
STD24044,RALAMBOMANANA,Fanomezantsoa Yohan
STD24046,ANDRIANTSOA,Lavorary David
STD24049,RAMANANDRAIBE,Lahatra Nomena
STD24051,ANDRIAMAMPIANINA,Toavina Ambinintsoa
STD24054,RATOVONJANAHARY,Rado Fanomezantsoa
STD24056,ANDRIAMAMPIONONA HARISON,Sitraka Mathieu
STD24059,BE,Warry Lionel
STD24061,RAVONJINIAINA,Eric Faneva Ankasitrahana
STD24062,RANDRIANANTENAINA,Nasandratra Nihando
STD24063,RANAIVOMANANA,Sombin'Ny Aina
STD24064,ANDRIANTSOA,Tafitaniaina Corina
STD24069,RASOLOFONDRASOA,Tsiry Rindraniaina
STD24072,RIANALA,Raminosata Mitsantafitiavana
STD24077,RATSIMANDRESY,Tojosoa Marc
STD24078,MANDANIAINA LALANIRINA,Miarintsoa Christian
STD24087,ANDRIANIHASINAVALONA,Tsanta Fitia Telina
STD24094,RAKOTOZANANY,Andry Tokiniaina
STD24111,RASAMOELY,Maminiaina Bryan
STD24117,NOROARILALAO,Nekena
STD24120,NANDRASANTSOA,Lantoharilala Sarindra
STD24127,RAHELIMANANA,Niavo Maminirina Marinah
STD24129,RADANIELSON,Ny Sedra Hasina Fabien
STD24131,MANANTENASOA,Fanaja
```

2 - Filtrer les lignes contenant "ANDRIA"

Pour afficher les lignes contenant le mot "ANDRIA", on peut utiliser différentes commandes grep :

- cat Liste_eleve | grep 'ANDRIA'

```
root@ubuntu:/home/rado/K3# cat Liste_eleve | grep 'ANDRIA'
STD23002,ANDRIAMARO,Andraina
STD24001,ANDRIANARIVOSON,Ny Aina Nathan
STD24004,RANDRIAMIZARA,Henintsoa Sarobidy Jessy
STD24006,RANDRIANIAINA,Diarilala
STD24024,RANDRIANASOLO,Ny Herin'ny Finoana
STD24046,ANDRIANTSOA,Lavorary David
STD24051,ANDRIAMAMPIANINA,Toavina Ambinintsoa
STD24056,ANDRIAMAMPIONONA HARISON,Sitraka Mathieu
STD24062,RANDRIANANTENAINA,Nasandratra Nihando
STD24064,ANDRIANTSOA,Tafitaniaina Corina
STD24087,ANDRIANIHASINAVALONA,Tsanta Fitia Telina
STD24144,ANDRIANAVALONA,Ny Tsiro
STD24155,ANDRIAMISAINA,Sitrakiniaina Tafika
STD24157,RANDRIARIMALALA,Zafindrazafy Claudine Élysée
```

- cat Liste_eleve | grep "ANDRIA"

```
root@ubuntu:/home/rado/K3# cat Liste_eleve | grep "ANDRIA"
STD23002,ANDRIAMARO,Andraina
STD24001,ANDRIANARIVOSON,Ny Aina Nathan
STD24004,RANDRIAMIZARA,Henintsoa Sarobidy Jessy
STD24006,RANDRIANIAINA,Diarilala
STD24024,RANDRIANASOLO,Ny Herin'ny Finoana
STD24046,ANDRIANTSOA,Lavorary David
STD24051,ANDRIAMAMPIANINA,Toavina Ambinintsoa
STD24056,ANDRIAMAMPIONONA HARISON,Sitraka Mathieu
STD24062,RANDRIANANTENAINA,Nasandratra Nihando
STD24064,ANDRIANTSOA,Tafitaniaina Corina
STD24087,ANDRIANIHASINAVALONA,Tsanta Fitia Telina
STD24144,ANDRIANAVALONA,Ny Tsiro
STD24155,ANDRIAMISAINA,Sitrakiniaina Tafika
STD24157,RANDRIARIMALALA,Zafindrazafy Claudine Élysée
```

- cat Liste_eleve | grep ANDRIA

```
root@ubuntu:/home/rado/K3# cat Liste_eleve | grep ANDRIA
STD23002,ANDRIAMARO,Andraina
STD24001,ANDRIANARIVOSON,Ny Aina Nathan
STD24004,RANDRIAMIZARA,Henintsoa Sarobidy Jessy
STD24006,RANDRIANIAINA,Diarilala
STD24024,RANDRIANASOLO,Ny Herin'ny Finoana
STD24046,ANDRIANTSOA,Lavorary David
STD24051,ANDRIAMAMPIANINA,Toavina Ambinintsoa
STD24056,ANDRIAMAMPIONONA HARISON,Sitraka Mathieu
STD24062,RANDRIANANTENAINA,Nasandratra Nihando
STD24064,ANDRIANTSOA,Tafitaniaina Corina
STD24087,ANDRIANIHASINAVALONA,Tsanta Fitia Telina
STD24144,ANDRIANAVALONA,Ny Tsiro
STD24155,ANDRIAMISAINA,Sitrakiniaina Tafika
STD24157,RANDRIARIMALALA,Zafindrazafy Claudine Élysée
```

- grep 'ANDRIA' Liste_eleve

```
root@ubuntu:/home/rado/K3# grep 'ANDRIA' Liste_eleve
STD23002,ANDRIAMARO,Andraina
STD24001,ANDRIANARIVOSON,Ny Aina Nathan
STD24004,RANDRIAMIZARA,Henintsoa Sarobidy Jessy
STD24006,RANDRIANIAINA,Diarilala
STD24024,RANDRIANASOLO,Ny Herin'ny Finoana
STD24046,ANDRIANTSOA,Lavorary David
STD24051,ANDRIAMAMPIANINA,Toavina Ambinintsoa
STD24056,ANDRIAMAMPIONONA HARISON,Sitraka Mathieu
STD24062,RANDRIANANTENAINA,Nasandratra Nihando
STD24064,ANDRIANTSOA,Tafitaniaina Corina
STD24087,ANDRIANIHASINAVALONA,Tsanta Fitia Telina
STD24144,ANDRIANAVALONA,Ny Tsiro
STD24155,ANDRIAMISAINA,Sitrakiniaina Tafika
STD24157,RANDRIARIMALALA,Zafindrazafy Claudine Élysée
```

- grep "ANDRIA" Liste_eleve

```
root@ubuntu:/home/rado/K3# grep "ANDRIA" Liste_eleve
STD23002,ANDRIAMARO,Andraina
STD24001,ANDRIANARIVOSON,Ny Aina Nathan
STD24004,RANDRIAMIZARA,Henintsoa Sarobidy Jessy
STD24006,RANDRIANIAINA,Diarilala
STD24024,RANDRIANASOLO,Ny Herin'ny Finoana
STD24046,ANDRIANTSOA,Lavorary David
STD24051,ANDRIAMAMPIANINA,Toavina Ambinintsoa
STD24056,ANDRIAMAMPIONONA HARISON,Sitraka Mathieu
STD24062,RANDRIANANTENAINA,Nasandratra Nihando
STD24064,ANDRIANTSOA,Tafitaniaina Corina
STD24087,ANDRIANIHASINAVALONA,Tsanta Fitia Telina
STD24144,ANDRIANAVALONA,Ny Tsiro
STD24155,ANDRIAMISAINA,Sitrakiniaina Tafika
STD24157,RANDRIARIMALALA,Zafindrazafy Claudine Élysée
```

- grep ANDRIA Liste_eleve

```
root@ubuntu:/home/rado/K3# grep ANDRIA Liste_eleve
STD23002,ANDRIAMARO,Andraina
STD24001,ANDRIANARIVOSON,Ny Aina Nathan
STD24004,RANDRIAMIZARA,Henintsoa Sarobidy Jessy
STD24006,RANDRIANIAINA,Diarilala
STD24024,RANDRIANASOLO,Ny Herin'ny Finoana
STD24046,ANDRIANTSOA,Lavorary David
STD24051,ANDRIAMAMPIANINA,Toavina Ambinintsoa
STD24056,ANDRIAMAMPIONONA HARISON,Sitraka Mathieu
STD24062,RANDRIANANTENAINA,Nasandratra Nihando
STD24064,ANDRIANTSOA,Tafitaniaina Corina
STD24087,ANDRIANIHASINAVALONA,Tsanta Fitia Telina
STD24144,ANDRIANAVALONA,Ny Tsiro
STD24155,ANDRIAMISAINA,Sitrakiniaina Tafika
STD24157,RANDRIARIMALALA,Zafindrazafy Claudine Élysée
```

Les trois premières commandes affichent le contenu du fichier avant de le filtrer avec grep, tandis que les trois dernières utilisent directement grep sur le fichier. On peut aussi utiliser des guillemets simples ou doubles, ou même pas de guillemets du tout, cela fonctionne pareil.

3 - Remplacer les virgules par des espaces

Une fois qu'on a filtré les lignes contenant "ANDRIA", on va remplacer les virgules par des espaces avec la commande sed :

sed 's/,/ /g'

Cette commande remplace toutes les virgules par des espaces dans le résultat.

```
root@ubuntu:/home/rado/K3# cat Liste_eleve | grep ANDRIA | sed 's/,/ /g'
STD23002 ANDRIAMARO Andraina
STD24001 ANDRIANARIVOSON Ny Aina Nathan
STD24004 RANDRIAMIZARA Henintsoa Sarobidy Jessy
STD24006 RANDRIANIAINA Diarilala
STD24024 RANDRIANASOLO Ny Herin'ny Finoana
STD24046 ANDRIANTSOA Lavorary David
STD24051 ANDRIAMAMPIANINA Toavina Ambinintsoa
STD24056 ANDRIAMAMPIONONA HARISON Sitraka Mathieu
STD24062 RANDRIANANTENAINA Nasandratra Nihando
STD24064 ANDRIANTSOA Tafitaniaina Corina
STD24087 ANDRIANIHASINAVALONA Tsanta Fitia Telina
STD24144 ANDRIANAVALONA Ny Tsiro
STD24155 ANDRIAMISAINA Sitrakiniaina Tafika
STD24157 RANDRIARIMALALA Zafindrazafy Claudine Élysée
```

4 - Filtrer les colonnes avec awk

Après avoir remplacé les virgules par des espaces, on peut utiliser la commande awk pour afficher uniquement le nom et le premier prénom. Comme awk sépare les colonnes par des espaces par défaut, cela fonctionne bien après le remplacement des virgules. La commande pour afficher le nom (colonne 2) et le prénom (colonne 3) est :

awk '{print $2, $3}'

```
root@ubuntu:/home/rado/K3# cat Liste_eleve | grep ANDRIA | sed 's/,/ /g' | awk '{print $2, $3}'
ANDRIAMARO Andraina
ANDRIANARIVOSON Ny
RANDRIAMIZARA Henintsoa
RANDRIANIAINA Diarilala
RANDRIANASOLO Ny
ANDRIANTSOA Lavorary
ANDRIAMAMPIANINA Toavina
ANDRIAMAMPIONONA HARISON
RANDRIANANTENAINA Nasandratra
ANDRIANTSOA Tafitaniaina
ANDRIANIHASINAVALONA Tsanta
ANDRIANAVALONA Ny
ANDRIAMISAINA Sitrakiniaina
RANDRIARIMALALA Zafindrazafy
```

Ou bien :

awk '{print $2 " " $3}'

```
root@ubuntu:/home/rado/K3# cat Liste_eleve | grep ANDRIA | sed 's/,/ /g' | awk '{print $2 " " $3}'
ANDRIAMARO Andraina
ANDRIANARIVOSON Ny
RANDRIAMIZARA Henintsoa
RANDRIANIAINA Diarilala
RANDRIANASOLO Ny
ANDRIANTSOA Lavorary
ANDRIAMAMPIANINA Toavina
ANDRIAMAMPIONONA HARISON
RANDRIANANTENAINA Nasandratra
ANDRIANTSOA Tafitaniaina
ANDRIANIHASINAVALONA Tsanta
ANDRIANAVALONA Ny
ANDRIAMISAINA Sitrakiniaina
RANDRIARIMALALA Zafindrazafy
```

Dans ces deux commandes, $2 représente la deuxième colonne (le nom), et $3 représente la troisième colonne (le prénom). Le " " ou la virgule entre $2 et $3 permet d'ajouter un espace entre les deux.

Commandes de fond :

L'**opérateur &** permet d'exécuter une commande en arrière-plan. Cela signifie que le processus fonctionne pendant que tu peux continuer à utiliser ton terminal pour d'autres commandes.

On va créer deux script (script1.sh et script2.sh) qui affiche HEI1 et HEI2 en boucle infinie

```
  GNU nano 4.8
#!/bin/bash
while true; do
  echo "HEI1"
  sleep 5
done
```

```
  GNU nano 4.8
#!/bin/bash
while true; do
  echo "HEI2"
  sleep 5
done
```

En ajoutant & à la fin de la commande, le script s'exécute en arrière-plan, ce qui permet à ton terminal de rester libre pour d'autres commandes.

```
root@ubuntu:/home/rado/K3# ./script1.sh &
```

Une fois que tu as lancé un processus en arrière-plan, tu peux le gérer avec certaines commandes. Voici ce que ces commandes font :

- **jobs** : Liste tous les processus en arrière-plan dans ton terminal.
- **fg** : Ramène un processus en avant-plan, c'est-à-dire qu'il occupe à nouveau ton terminal.
- **bg** : Si un processus est arrêté, tu peux le reprendre et le mettre en arrière-plan.

On va lancer la commande jobs qui liste les processus en arrière-plan avec un numéro pour chaque job. :

Les numéros [1] et [2] sont les identifiants des jobs. Les numéros 12345 et 12346 sont les identifiants des processus.

```
root@ubuntu:/home/rado/K3# jobs
[1]-  Running                 ./script1.sh &
[2]+  Running                 ./script2.sh &
```

Pour ramener script1.sh en avant-plan (occupant ton terminal), on utilise **fg** suivi du numéro du job (ici 1 pour le premier job) :

```
root@ubuntu:/home/rado/K3# fg 1
./script1.sh
```

Cela ramène le script script1.sh au premier plan, et tu verras les sorties HEI1 directement dans le terminal.

Planification des tâches avec cron :

Dans cette partie, on va planifier une tâche que le système va exécuter toutes les minutes. Voici les étapes :

1. **Créer un fichier appelé "cron"** :

   D'abord, on va créer un fichier qu'on va appeler cron où on enregistrera le résultat de notre commande.

   ```
   root@ubuntu:/home/rado/K3# touch cron
   ```

2. **Créer un script** :

   On va créer un script qui exécute la commande date. Cette commande va afficher la date du jour, avec l'heure et la minute. Le résultat sera ensuite enregistré dans le fichier cron que l'on vient de créer.

   ```
     GNU nano 4.8
   #!/bin/bash
   date '+%D %H%M' >> /home/rado/K3/cron
   ```

3. **Planifier l'exécution avec Cron** :

Ensuite, on va utiliser Cron pour que le système lance notre script toutes les minutes.

```
  GNU nano 4.8
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# 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
 * * * * * cd /home/rado/K3 && ./cron.sh
```

Cette ligne signifie que le script cron.sh sera exécuté toutes les minutes, dans le répertoire /home/rado/K3.

4. **Résultat**

```
root@ubuntu:/home/rado/K3# cat cron
03/13/25 0848
03/13/25 0849
03/13/25 0850
03/13/25 0851
03/13/25 0852
03/13/25 0853
```

Automatisation et scripts

Création d'un script de sauvegarde automatisé :

Dans cette partie, on va créer un système de sauvegarde avec un script simple, sans boucle. Voici ce que fera le script :

1. On va créer un dossier pour la sauvegarde, et pour plus de sécurité, chaque jour le dossier aura un nom différent. Le nom du dossier sera basé sur la date du jour (par exemple, 1303 pour le 13 mars).
2. On va limiter l'accès au dossier pour que seul le créateur (ici l'utilisateur root) puisse y accéder, en utilisant la commande chmod.
3. Ensuite, on va copier tous les fichiers journaux du système (qui ont plus de 24h) du répertoire /var/log vers ce dossier sécurisé. Après avoir copié les fichiers, on les supprimera.

Maintenant pour créer une variable **nom_dossier** qui contiendra le nom du dossier basé sur la date du jour :

nom_dossier=$(date '+%d%m')

Cela crée une variable avec le nom du dossier basé sur la date (par exemple, 1303 pour le 13 mars).

Pour changer les permissions du dossier, on va protéger le dossier en enlevant les permissions pour tous les utilisateurs autres que le créateur (root) avec la commande **chmod** :

chmod go-rwx $nom_dossier

Cela supprime les permissions pour "groupe" et "autres" utilisateurs, permettant seulement au créateur (root) d'y accéder.

Pour rechercher les fichiers réguliers datant de plus de 24 heures dans le répertoire /var/log :

find /var/log -type f -mtime +24

-type f signifie que nous cherchons des fichiers réguliers.

-mtime +24 signifie que nous cherchons des fichiers qui ont été modifiés il y a plus de 24 heures.

Pour chaque fichier trouvé, on exécute la commande cp -f pour copier ce fichier dans le répertoire de sauvegarde ($nom_dossier), en écrasant les fichiers existants si nécessaire.

find /var/log -type f -mtime +24 -exec cp -f {} $nom_dossier \;

- L'option -**exec** permet d'exécuter une commande sur chaque fichier trouvé par find. Cela signifie que pour chaque fichier que find trouve, la commande suivante (ici, cp -f) sera exécutée sur ce fichier.
- **cp** est la commande utilisée pour copier des fichiers.

- -f signifie **"force"**. Cela permet de copier un fichier, même s'il existe déjà un fichier avec le même nom dans le répertoire de destination. La commande écrasera ce fichier sans demander de confirmation.
- Les **accolades {}** représentent le fichier trouvé par find. Pour chaque fichier que find trouve, {} sera remplacé par le chemin complet du fichier.
- $nom_dossier est la variable que tu as définie plus tôt dans le script, qui contient le nom du dossier de sauvegarde (basé sur la date du jour, comme 1303 pour le 13 mars).
- **\;** marque la fin de la commande -exec. C'est nécessaire pour indiquer à find que la commande à exécuter est terminée.

Une fois que les données ont été copiées avec la commande find précédente, on va maintenant les supprimer du répertoire /var/log avec la commande suivante :

find /var/log -type f -mtime +24 -exec rm -f {} \;

Cette commande fonctionne de la même manière que la précédente, mais au lieu de copier les fichiers, elle supprime ceux qui ont plus de 24 heures (avec la commande rm -f).

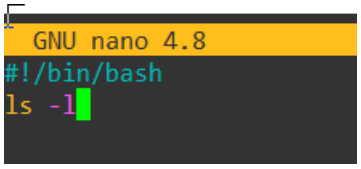
Voici le script complet :

```
  GNU nano 4.8
#!/bin/bash
echo "Creation du repertoire de sauvegarde"
nom_dossier=$(date '+%d%m')
        mkdir $nom_dossier
# Modification de la permission du repertoire
        chmod go-rwx $nom_dossier
echo "sauvegarde des fichiers log de plus de 24h"
        find /var/log -type f -mtime +24 -exec cp -f {} $nom_dossier \;
echo "Suppression fichier log plus de 24 heures"
        find /var/log -type f -mtime +24 -exec rm -f {} \;
```

Transformation d'un script en commande exécutable globale :

Pour transformer ton script en commande, il suffit de le copier dans le dossier **/usr/local/bin**. Par exemple, imaginons qu'on crée une commande personnalisée appelée **listeo** qui exécute la commande ls -l.

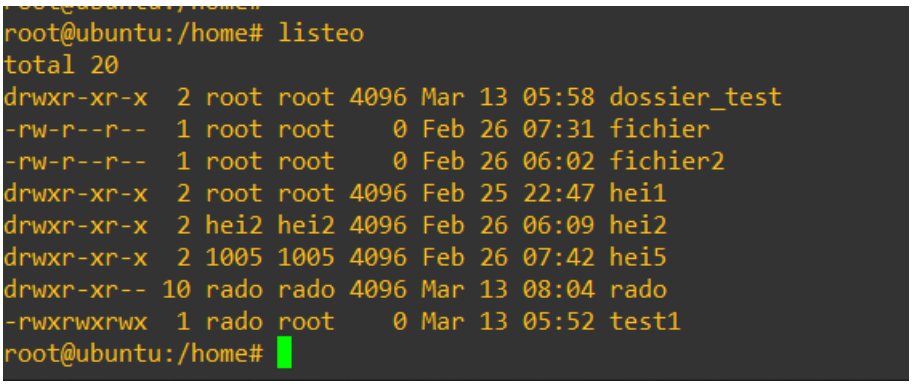
Création de notre script avec le contenu suivant :



Copie du script dans /usr/local/bin :

sudo cp listeo /usr/local/bin

Résultat : maintenant, on peut utiliser **listeo** comme n'importe quelle autre commande dans le terminal !



Système de surveillance :

Dans cette partie, on va créer un script pour surveiller notre système (processeur, mémoire et stockage). On va utiliser les commandes qu'on a vues en cours. Voici comment ça va se passer :

1. **Variable 1 : utilisation_cpu**

   Cette variable va récupérer l'info sur l'utilisation du processeur. On va utiliser la commande top, puis extraire la ligne contenant "Cpu(s)".

   Ensuite, **awk '{print $2 + $4 + $6 + $10}'** qui extrait et additionne les pourcentages d'utilisation du CPU, en tenant compte de :

   $2 : Le pourcentage d'usage utilisateur (us).

   $4 : Le pourcentage d'usage système (sy).

   $6 : Le pourcentage d'usage de l'IO (wa).

   $10 : Le pourcentage d'usage d'autres tâches comme l'irruption (st).

   Variable 2 : information_memoire

   Cette variable va récupérer les infos sur la mémoire. On utilise la commande free et on filtre la ligne contenant "Mem". Cela nous donnera les données nécessaires sur la mémoire RAM.

2. **Variable 3 : memoire_total**

   Cette variable va récupérer la deuxième colonne de la variable précédente, qui correspond à la quantité totale de mémoire.

3. **Variable 4 : memoire_utilise**

   Cette variable va récupérer la troisième colonne de la même sortie, qui indique la mémoire utilisée.

4. **Variable 5 : memoire_libre**

   Cette variable va récupérer la quatrième colonne de la sortie, qui donne la mémoire libre.

5. **Variable 6 : utilisation_stockage**

   Cette variable va récupérer les infos sur le stockage avec la commande df. On extrait la ligne contenant "dev", puis on récupère les colonnes qui montrent l'utilisation du disque, la partition, l'espace utilisé et l'espace total.

```
 GNU nano 4.8                                                    monitoring.sh
#!/bin/bash

# Boucle infinie
while true; do
    # Affichage de la date et de l'heure pour savoir quand la mise à jour a eu lieu
    clear
    echo "---------------------------------"
    echo "Statistiques Système en Temps Réel"
    echo "---------------------------------"
    echo "Date et Heure : $(date)"
    echo ""

    # Affichage de l'utilisation du processeur en pourcentage
    utilisation_cpu=$(top -bn1 | grep "Cpu(s)" | awk '{print $2 + $4 + $6 + $10}')
    echo "Utilisation du Processeur : $utilisation_cpu%"

    # Affichage de l'utilisation de la mémoire
    information_memoire=$(free -h | grep Mem)
    memoire_total=$(echo $information_memoire | awk '{print $2}')
    memoire_utilise=$(echo $information_memoire | awk '{print $3}')
    memoire_libre=$(echo $information_memoire | awk '{print $4}')
    echo "Mémoire : Utilisée $memoire_utilise / Libre $memoire_libre / Totale $memoire_total"

    # Affichage de l'utilisation du stockage (disque dur)
    utilisation_stockage=$(df -h | grep '^/dev' | awk '{print $5 " sur " $1 " (" $3 " utilisé sur " $2 ")"}')
    echo "Stockage :"
    echo "$utilisation_stockage"

    # Pause de 2 secondes avant la prochaine mise à jour
    sleep 2
done
```

Résultat :

root@ubuntu: /home/rado/K3

```
---------------------------------
Statistiques Système en Temps Réel
---------------------------------
Date et Heure : Sat 15 Mar 2025 10:18:22 AM UTC

Utilisation du Processeur : 8.2%
Mémoire : Utilisée 422Mi / Libre 5.6Gi / Totale 7.7Gi
Stockage :
63% sur /dev/mapper/ubuntu--vg-ubuntu--lv (12G utilisé sur 20G)
24% sur /dev/sda2 (214M utilisé sur 974M)
100% sur /dev/loop0 (56M utilisé sur 56M)
100% sur /dev/loop2 (64M utilisé sur 64M)
100% sur /dev/loop1 (56M utilisé sur 56M)
100% sur /dev/loop3 (64M utilisé sur 64M)
100% sur /dev/loop5 (45M utilisé sur 45M)
100% sur /dev/loop4 (92M utilisé sur 92M)
100% sur /dev/loop6 (92M utilisé sur 92M)
```