Chapitre 4

Gestion des processus

Résumé Détaillé du Cours : Gestion des Processus sous Linux

I. Concepts Fondamentaux

1. Définition d'un Processus

- **Programme** = Suite d'instructions (objet statique)
- **Processus** = Programme en exécution + Contexte (objet dynamique)
- **Contexte/Environnement** comprend :
  - Code
  - Données temporaires et permanentes
  - Fichiers associés
  - Variables
  - Descripteurs
  - Mémoire allouée
  - Pile d'exécution

2. Types de Processus

1. **Processus d'avant-plan (foreground)**
   - Interagit avec l'utilisateur
   - Exemple : éditeur de texte
2. **Processus d'arrière-plan (background)**
   - Sans interaction utilisateur
   - Exemple : démons (daemons)

3. États d'un Processus

1. **Actif (Run)**
   - En cours d'exécution sur le CPU
2. **Prêt**
   - En attente de l'ordonnanceur
3. **Endormi (Sleep)**
   - En attente d'un événement

    - Types :
        - Sleep normal
        - Disk Sleep (attente d'E/S)
4. **Suspendu (Stopped)**
    - Interrompu par un signal
5. **Zombie**
    - État instable
    - Processus planté utilisant encore la mémoire

II. Commandes Essentielles de Gestion des Processus

1. Visualisation des Processus

A. `pstree`

Fonction : Affiche l'arbre des processus

```
bash
Copy
pstree                  # Affiche tous les processus
pstree [utilisateur]    # Processus d'un utilisateur spécifique
```

B. `top`

Fonction : Moniteur système en temps réel

```
bash
Copy
top    # Lance le moniteur système
```

Options interactives :

- `h` : Aide
- `L` : Recherche
- `f` : Modification de l'affichage
- `q` : Quitter
- `r` : Modifier la priorité

Informations affichées :

1. Première ligne :
    - Heure
    - Uptime
    - Utilisateurs connectés
    - Charge système
2. Deuxième ligne :

- Nombre total de processus
- États des processus (running, sleeping, stopped, zombie)

3. Troisième ligne : Usage CPU
   - us : temps utilisateur
   - sy : temps système
   - id : temps idle
   - wa : temps d'attente I/O

C. `ps`

Fonction : État des processus

bash
Copy
```bash
ps        # Processus de la session
ps -A     # Tous les processus
ps -e     # Équivalent à -A
```

2. Contrôle des Processus

A. Gestion des Jobs

bash
Copy
```bash
jobs    # Liste les tâches en arrière-plan
fg %n   # Passe la tâche n en avant-plan
bg %n   # Passe la tâche n en arrière-plan
```

B. Identification des Processus

bash
Copy
```bash
pidof nom_processus    # Trouve le PID d'un processus
```

3. Gestion de la Priorité

A. `nice` et `renice`

Échelle : -20 (priorité maximale) à 19 (priorité minimale)

bash
Copy
```bash
# Modification de priorité
renice priorité -p PID           # Pour un processus
renice priorité -u utilisateur   # Pour un utilisateur
```

4. Signaux et Terminaison

A. `kill`

Fonction : Envoie des signaux aux processus

```
bash
Copy
kill PID          # Terminaison normale
kill -9 PID       # Terminaison forcée
kill -signal PID  # Signal spécifique
```

Signaux principaux :

1. SIGHUP (1) : Fin de session
2. SIGINT (2) : Interruption
3. SIGKILL (9) : Terminaison immédiate
4. SIGTERM (15) : Terminaison normale
5. SIGSTOP (19) : Suspension
6. SIGCONT (18) : Reprise d'exécution

III. Bonnes Pratiques

1. Gestion des Processus

- Préférer la terminaison normale (SIGTERM)
- N'utiliser SIGKILL qu'en dernier recours
- Vérifier régulièrement l'état des processus

2. Performance

- Surveiller la charge système avec `top`
- Gérer les priorités selon les besoins
- Éviter l'accumulation de processus zombies

3. Sécurité

- Limiter l'utilisation des droits root
- Surveiller les processus inconnus
- Nettoyer régulièrement les processus inutiles

IV. Points Clés pour l'Examen

1. Comprendre les états des processus
2. Maîtriser la lecture de `top`
3. Connaître les signaux principaux
4. Savoir gérer les priorités
5. Comprendre la différence avant-plan/arrière-plan
6. Maîtriser les commandes de base (ps, top, kill)
7. Comprendre le concept de PID et PPID

Corrigé TD/TP 3 - Linux

Exercice 1 : Entête et queue

1. Affichage des lignes du fichier /etc/hosts :

bash
Copy

```
# 15 premières lignes
head -n 15 /etc/hosts

# 15 dernières lignes
tail -n 15 /etc/hosts

# À partir de la 15ème ligne
tail -n +15 /etc/hosts

# Lignes 15 à 20
head -n 20 /etc/hosts | tail -n 6
```

2. Pour récupérer les lignes 5 à 9 d'un fichier :

bash
Copy

```
head -n 9 fichier | tail -n 5
```

3. Pour afficher la 5ème ligne uniquement :

bash
Copy

```
head -n 5 fichier | tail -n 1
```

Exercice 2: Filtres et redirections

1. Création et manipulation du fichier :

bash
Copy

```
# Création du fichier (exemple)
cat > fichier.txt << EOF
1 / premier mot
2 / deuxieme mot
3 / troisieme mot
EOF

# a) Tri numérique
sort -n fichier.txt

# b) Éliminer le chiffre et le slash
cut -d' ' -f3- fichier.txt
```

```
# c) Tri alphabétique inverse
cut -d' ' -f3- fichier.txt | sort -r
```

2. Pour compter les fichiers de configuration (cachés) :

bash
Copy
```
ls -la ~ | grep "^-.*\." | grep -v "~$" | wc -l
```

3. Pour compter les répertoires de configuration :

bash
Copy
```
ls -la ~ | grep "^d.*\." | wc -l
```

4. Pour lister tous les fichiers de l'arborescence :

bash
Copy
```
ls -R > liste_fichiers.txt
```

5. Création d'un fichier contenant la liste détaillée :

bash
Copy
```
ls -l > liste
```

6. Afficher les liens symboliques du répertoire courant :

bash
Copy
```
ls -l | grep "^l"
```

7. Compter les occurrences de "file" dans le man :

bash
Copy
```
man syst | grep -c "file"
```

8. Les 10 plus gros fichiers de /usr/bin :

bash
Copy
```
ls -lS /usr/bin | head -n 10
```

9. Pour le fichier /etc/hosts :

bash
Copy
```
# a) Cinquième caractère
```

```
cut -c5 /etc/hosts

# b) Caractères 5 à 10 et 13
cut -c5-10,13 /etc/hosts

# c) À partir du 15ème caractère
cut -c15- /etc/hosts
```

Exercice 3 : Script Shell

bash
Copy

```
#!/bin/bash

# Création du script
echo "mon nom est $0"
echo "je suis appele avec $# arguments"
echo -n "qui sont: "
for arg in "$@"
do
    echo -n "$arg "
done
echo ""

# Pour exécuter le script :
# chmod +x script.sh
# ./script.sh "Bienvenue" "dans" "le" "monde" "Linux"
# ./script.sh "Bienvenue dans le monde Linux"
```

Explications supplémentaires :

- `$0` : nom du script
- `$#` : nombre d'arguments
- `$@` : liste des arguments
- Le script utilise une boucle for pour parcourir tous les arguments
- L'option `-n` avec echo empêche le retour à la ligne

Pour tester avec les arguments demandés :

bash
Copy

```
./script.sh Bienvenue dans le monde Linux
./script.sh "Bienvenue dans le monde Linux"
```

Ces solutions utilisent principalement les commandes de base Linux comme :

- head et tail pour la manipulation des lignes
- grep pour la recherche
- cut pour l'extraction de caractères
- sort pour le tri

- ls pour le listage
- wc pour le comptage
- Les redirections (>, >>) et les pipes (|)