Systèmes d'exploitation centralisés

# Rapport : minishell

1SN-F


***Élève :***

**Louis Thevenet**

***Enseignant :***

**Emannuel Chaput**


4 Mai 2024

# Table des matières

# 1. Gestion des processus

## 1.1. Enchaînement séquentiel des commandes

```
> ls
flake.lock  flake.nix  projet  rapport  result  sujets  tp
> echo test
test
> cat ./projet/test
exemple
> exit
Au revoir ...
```

Liste 1. – On attend la fin de l'éxecution du fils pour passer à la prochaine commande

## 1.2. Exécution en arrière-plan

```
> cat ./projet/test
#!/usr/bin/env bash

sleep 5
echo "Done!"
> ./projet/test &
>
> ls
flake.lock  flake.nix  projet  rapport  result  sujets  tp
> Done!
>
```

Liste 2. – On exécute la commande en arrière-plan

On peut également vérifier la bonne terminaison du fils après éxecution via `watch ps -sf` :

```
S+   pts/2      0:00      \_ /nix/store/cci0aml5v6xdvkqrvg-minishell/bin/minishell
S+   pts/2      0:00          \_ bash ./projet/test
S+   pts/2      0:00              \_ sleep 5
```

Liste 3. – Durant l'exécution

```
S+   pts/2      0:00      \_ /nix/store/cci0aml5v6xdvkqrvg-minishell/bin/minishell
```

Liste 4. – Après exécution

# 2. Signaux

## 2.1. Signal `SIGCHLD`

Un mode `debug` a été ajouté au projet afin d'afficher des informations sur les signaux reçus.

Dans la Liste 5, on :

- attend normalement la fin d'éxecution de la commande en avant-plan
- attend normalement la fin d'éxecution de la commande en arrière-plan
- envoie le signal `SIGCHLD` au processus fils
- envoie le signal `SIGSTOP` au processus fils
- envoie le signal `SIGCONT` au processus fils
- affiche les jobs en cours pour constater que le fils est continué en arrière-plan
- On teste ensuite les signaux `SIGSTOP` et `SIGCONT` sur un job en arrière-plan.

```
> sleep 2
[Child 294501 exited]
> sleep 2&
> [Child 294709 exited]

> sleep 9999
[Child 296018 signaled]
> sleep 9999
[Child 299625 stopped]
> [Child 299625 continued]

> lj
Job 0
  PID: 299625
STDOUT -> 4207432 State: Active Command: sleep 9999
>
>
>
>
>
> sleep 999&
> [Child 335139 stopped]
[Child 335139 continued]

> lj
Job 0
  PID: 299625
STDOUT -> 4207432 State: Active Command: sleep 9999

Job 1
  PID: 335139
STDOUT -> 4207432 State: Active Command: sleep 999
>
```

Liste 5. – Démonstration `SIGCHLD`

## 2.2. Signaux `SIGINT`, `SIGTSTP`

On voit dans cet exemple que le programme père reçoit le signal `SIGINT`, qu'il décide de tuer le fils en avant-plan, finalement le message informant la terminaison du processus fils est affiché.

```
> sleep 10
^C[SIGINT received]
[Killing 343628]
> [Child 343628 signaled]

>
```

Liste 6. – Interruption au clavier

```
> sleep 10
^Z[SIGTSTP received]
[Stopping 348897]
> [Child 348897 stopped]

> lj
Job 0
  PID: 348897
STDOUT -> 4207432 State: Suspended  Command: sleep 10
>
```

Liste 7. – Envoie du signal `SIGTSTP`

Les processus fils quant à eux ne reçoivent pas les signaux `SIGINT` et `SIGTSTP`, c'est le père qui gère les interruptions.

```
> sleep 999&
> ^C[SIGINT received]
^C[SIGINT received]
^Z[SIGTSTP received]
^Z[SIGTSTP received]

> lj
Job 0
  PID: 355672
STDOUT -> 4207432 State: Active Command: sleep 999
>
```

Liste 8. – Les processus fils masquent les signaux `SIGINT` et `SIGTSTP`

# 3. Fichiers et redirections

# 4. Tubes