Polynésie – 2023 – sujet2 - Correction

Exercice 3 (4 points)

1.

1.a. Hiérarchie des processus liés à firefox

À partir de l'affichage `ps -aef`, on repère :

- le processus firefox principal (PPID ≠ firefox)
- les processus enfants dont le PPID est celui du firefox principal
- éventuellement des sous-processus

La hiérarchie s'écrit sous forme arborescente :

```
PID_parent
└── PID_firefox
    ├── PID_fils1
    ├── PID_fils2
    └── ...
```

👉 *Commentaire : On utilise la colonne PPID pour relier chaque processus à son parent.*

1.b. Commande ayant lancé le premier processus firefox

La commande qui a lancé le premier processus de `firefox` est `bash`.

👉 *Commentaire : Le premier processus firefox est celui dont le parent n'est pas un autre processus firefox.*

1.c. Supprimer tous les processus liés à firefox

Commande possible :

`killall firefox` ou `kill 9617`

👉 *Commentaire : Ces commandes suppriment tous les processus portant le nom firefox.*

2.

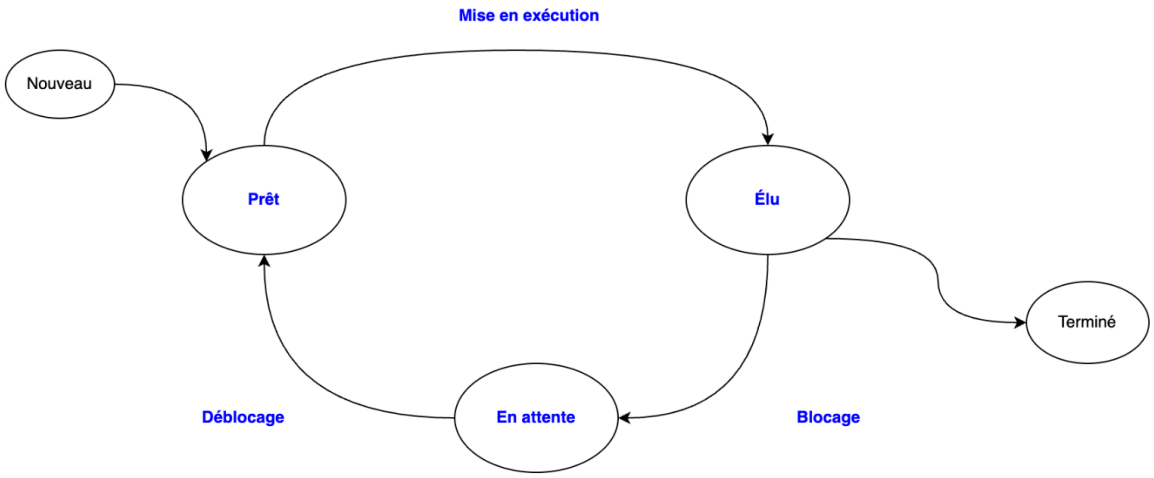
2.a. Schéma d'ordonnancement

États à placer :

- **Prêt**
- **Élu**
- **En attente**
- **Mise en exécution**
- **Blocage**
- **Déblocage**

Transitions :

- Prêt → Élu : Mise en exécution
- Élu → Prêt : Blocage
- Blocage → Prêt : Déblocage
- Prêt ↔ En attente selon disponibilité CPU



👉 *Commentaire : Il s'agit du schéma classique des états d'un processus.*

2.b. Ordonnancement (1 cycle à chaque fois)

Données

Processus	Arrivée	Durée
P1	0	12
P2	2	16
P3	3	2
P4	7	2

Algorithme : **Plus court temps restant (SRTF)** avec préemption à chaque cycle.

Ordonnancement obtenu :

```
0-3 : P1
3-5 : P3
5-7 : P1
7-9 : P4
9-13 : P2
13-16 : P1
```

Temps de fin :

- P1 → 12
- P2 → 18
- P3 → 5
- P4 → 9

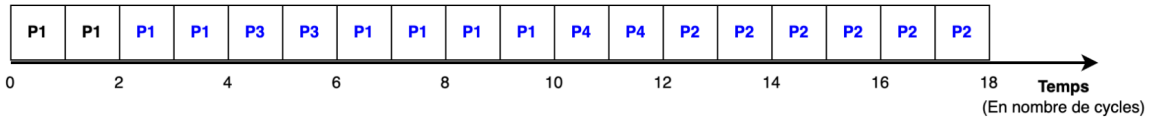
Temps d'exécution :

- P1 : 12 − 0 = 12
- P2 : 18 − 2 = 16
- P3 : 5 − 3 = 2
- P4 : 9 − 7 = 2

Moyenne : $\frac{12+16+2+2}{4} = 8$

👉 *Commentaire : L'algorithme favorise les processus courts.*

2.c. Nouvel ordonnancement des quatre processus



2.d. Nouvelle moyenne des temps d'exécution

Même principe mais exécution par blocs de 2 cycles.

Ordonnancement :

```
0-4 : P1
4-6 : P3
6-8 : P1
8-10 : P4
10-14 : P2
14-16 : P1
```

Temps de fin :

- P1 → 16
- P2 → 14
- P3 → 6
- P4 → 10

Temps d'exécution :

- P1 : 16
- P2 : 14 − 2 = 12
- P3 : 6 − 3 = 3
- P4 : 10 − 7 = 3

Moyenne : $\frac{16 + 12 + 3 + 3}{4} = 8,5$

👉 *Commentaire : La moyenne est plus élevée. Le premier ordonnancement est donc plus performant.*

3.

3.a. Fonction choix_processus

```
def choix_processus(liste_attente):
    if liste_attente != []:
        mini = len(liste_attente[0])
        indice = 0
        for i in range(1, len(liste_attente)):
            if len(liste_attente[i]) < mini:
                mini = len(liste_attente[i])
                indice = i
        return indice
```

👉 *Commentaire : On sélectionne la liste la plus courte (temps restant minimal).*

3.b. Fonction ordonnancement

```
def ordonancement(liste_proc):
    execution = []
    attente = scrutation(liste_proc, [])
    while attente != []:
        indice = choix_processus(attente)
        processus = attente[indice]
        execution.append(processus.pop(0))
        if len(processus) == 0:
            attente.pop(indice)
        attente = scrutation(liste_proc, attente)
    return execution
```

👉 *Commentaire :*
On retire un cycle (pop) du processus choisi.
Si le processus est vide, il est terminé.
On rescrute à chaque cycle.
