Exercices

Exercice Bac 2021

Cet exercice donné au bac en 2021 était sur 4 points. Le candidat devait choisir 3 exercices parmi 5 exercices proposés. Les sujets actuels ne contiennent que 3 exercices et dans ce contexte, je doute qu'un exercice entier porte sur les processus.

1. Les états possibles d'un processus sont : *prêt*, *élu*, *terminé* et *bloqué*.
a. Expliquer à quoi correspond l'état *élu*.
b. Proposer un schéma illustrant les passages entre les différents états.
2. On suppose que quatre processus C_1, C_2, C_3 et C_4 sont créés sur un ordinateur, et qu'aucun autre processus n'est lancé sur celui-ci, ni préalablement ni pendant l'exécution des quatre processus.
L'ordonnanceur, pour exécuter les différents processus prêts, les place dans une structure de données de type file. Un processus prêt est enfilé et un processus élu est défilé.
a. Parmi les propositions suivantes, recopier celle qui décrit le fonctionnement des entrées/sorties dans une file :

i. Premier entré, dernier sorti
ii. Premier entré, premier sorti
iii. Dernier entré, premier sorti

b. On suppose que les quatre processus arrivent dans la file et y sont placés dans l'ordre C_1, C_2, C_3 et C_4.
- Les temps d'exécution totaux de C_1, C_2, C_3 et C_4 sont respectivement 100 ms, 150 ms, 80 ms et 60 ms.
- Après 40 ms d'exécution, le processus C_1 demande une opération d'écriture disque, opération qui dure 200 ms. Pendant cette opération d'écriture, le processus C_1 passe à l'état bloqué.
- Après 20 ms d'exécution, le processus C_3 demande une opération d'écriture disque, opération qui dure 10 ms. Pendant cette opération d'écriture, le processus C_3 passe à l'état bloqué.

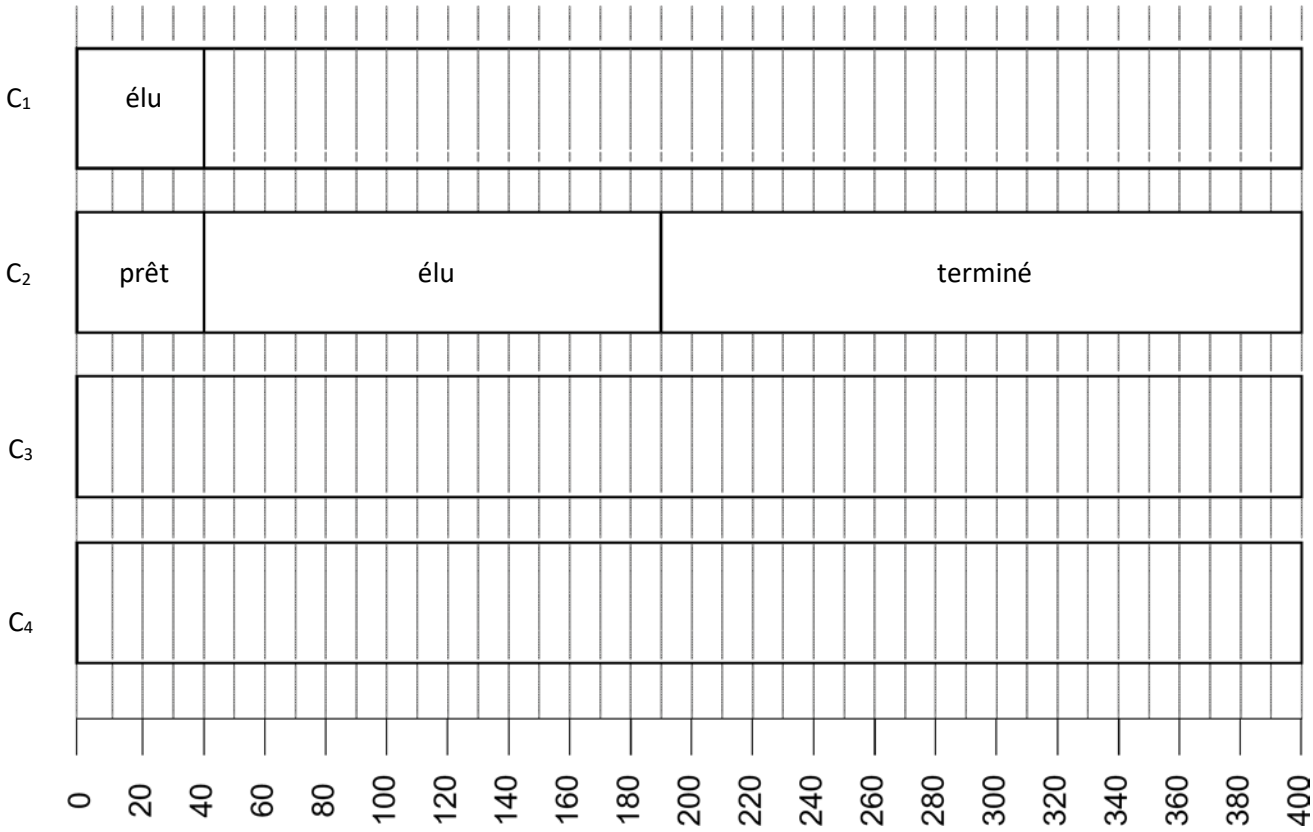
Sur la frise chronologique donnée en annexe (à rendre avec la copie), les états du processus C_2 sont donnés. Compléter la frise avec les états des processus C_1, C_3 et C_4.

3. On trouvera ci-dessous deux programmes rédigés en pseudo-code. Verrouiller un fichier signifie que le programme demande un accès exclusif au fichier et l'obtient si le fichier est disponible.

Programme 1	Programme 2
Verrouiller fichier_1	Verrouiller fichier_2
Calculs sur fichier_1	Verrouiller fichier_1
Verrouiller fichier_2	Calculs sur fichier_1
Calculs sur fichier_1	Calculs sur fichier_2
Calculs sur fichier_2	Déverrouiller fichier_1
Calculs sur fichier_1	Déverrouiller fichier_2
Déverrouiller fichier_2	
Déverrouiller fichier_1	

a. En supposant que les processus correspondant à ces programmes s'exécutent simultanément (exécution concurrente), expliquer le problème qui peut être rencontré.
b. Proposer une modification du programme 2 permettant d'éviter ce problème.

Annexe Question 2. b



Exercice Bac 2022

L'entreprise capNSI gère les contrats de ses clients en créant pour chacun d'eux un sous-dossier dans le dossier **Contrats** sur leur ordinateur central. Le système d'exploitation de cet ordinateur est une distribution linux. Quelques commandes de bases pour ce système d'exploitation sont rappelées en annexe 1 en fin de sujet.

Dans la console représentée sur la figure ci-dessous, on peut visualiser les répertoires (ou dossiers) à la racine de l'ordinateur central avec l'instruction **`ls :`**

```
gestion@capNSI-ordinateur_central:~$ ls
Bureau   Documents   Modèles   Public
Téléchargements Contrats   Images   Musique Vidéos
```

1. a. Donner le nom de l'utilisateur et le nom de l'ordinateur correspondant à la capture d'écran précédente.

b. Ecrire les instructions permettant d'afficher la liste des dossiers clients du répertoire `Contrats` en partant de la situation ci-dessous :

```
gestion@capNSI-ordinateur_central:~$
```

Après une campagne de démarchage, l'entreprise a gagné un nouveau client, Monsieur Alan Turing. Elle souhaite lui créer un sous-dossier nommé **`TURING_Alan`** dans le dossier **`Contrats`**. De plus, elle souhaite attribuer tous les droits à l'utilisateur et au groupe et seulement la permission en lecture pour tous les autres utilisateurs. La commande **`chmod`** permet de le faire.

2. a. Ecrire les instructions permettant de créer le sous-dossier **`TURING_Alan`** à partir du répertoire racine.

b. Ecrire l'instruction permettant d'attribuer les bons droits au sous-dossier **`TURING_Alan`**.

EXERCICE 1 (6 points) bac 2024

Cet exercice porte sur la programmation Python, la programmation orientée objet, les structures de données (file), l'ordonnancement et l'interblocage.

On s'intéresse aux processus et à leur ordonnancement au sein d'un système d'exploitation. On considère ici qu'on utilise un monoprocesseur.

1. Citer les trois états dans lesquels un processus peut se trouver.

On veut simuler cet ordonnancement avec des objets. Pour ce faire, on dispose déjà de la classe `Processus` dont voici la documentation :

Classe Processus:

```
p = Processus(nom: str, duree: int)
    Crée un processus de nom <nom> et de durée <duree> (exprimée en
cycles d'ordonnancement)

p.execute_un_cycle()
    Exécute le processus donné pendant un cycle.

p.est_fini()
    Renvoie True si le processus est terminé, False sinon.
```

Pour simplifier, on ne s'intéresse pas aux ressources qu'un processus pourrait acquérir ou libérer.

2. Citer les deux seuls états possibles pour un processus dans ce contexte.

Pour mettre en place l'ordonnancement, on décide d'utiliser une file, instance de la classe `File` ci-dessous.

Classe File

```
1  class File:
2      def __init__(self):
3          """ Crée une file vide """
4          self.contenu = []
5
6      def enfile(self, element):
7          """ Enfile element dans la file """
8          self.contenu.append(element)
9
10     def defile(self):
11         """ Renvoie le premier élément de la file et l'enlève de
la file """
12         return self.contenu.pop(0)
13
14     def est_vide(self):
```

```
15          """ Renvoie True si la file est vide, False sinon """
16          return self.contenu == []
```

Lors de la phase de tests, on se rend compte que le code suivant produit une erreur :

```
1 f = File()
2 print(f.defile())
```

3. Rectifier sur votre copie le code de la classe `File` pour que la function `defile` renvoie `None` lorsque la file est vide.

On se propose d'ordonnancer les processus avec une méthode du type *tourniquet* telle qu'à chaque cycle :

- si un nouveau processus est créé, il est mis dans la file d'attente ;
- ensuite, on défile un processus de la file d'attente et on l'exécute pendant un cycle ;
- si le processus exécuté n'est pas terminé, on le replace dans la file.

Par exemple, avec les processus suivants

Liste des processus		
processus	cycle de création	durée en cycles
A	2	3
B	1	4
C	4	3
D	0	5

On obtient le chronogramme ci-dessous :

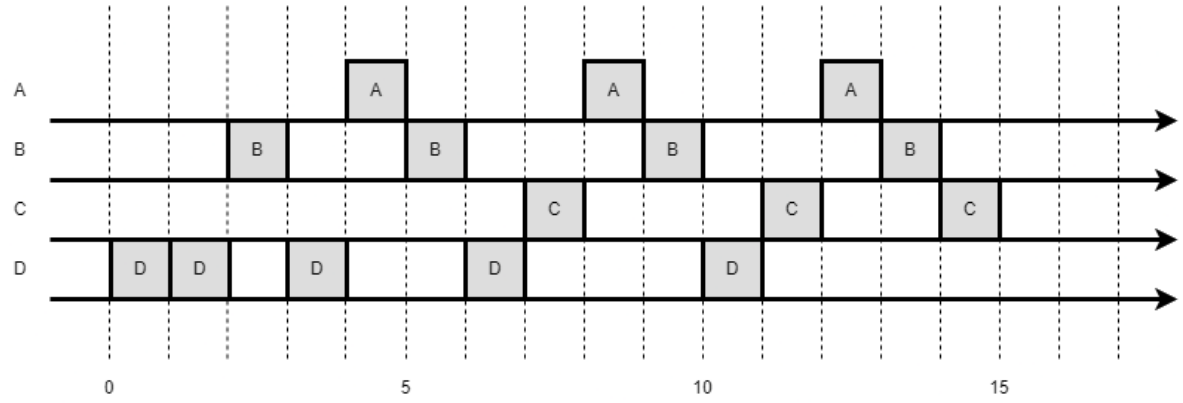


Figure 1. Chronogramme pour les processus A, B, C et D

Pour décrire les processus et le moment de leur création, on utilise le code suivant, dans lequel `depart_proc` associe à un cycle donné le processus qui sera créé à ce moment :

```
1 p1 = Processus("p1", 4)
2 p2 = Processus("p2", 3)
3 p3 = Processus("p3", 5)
4 p4 = Processus("p4", 3)
5 depart_proc = {0: p1, 1: p3, 2: p2, 3: p4}
```

Il s'agit d'une modélisation de la situation précédente où un seul processus peut être créé lors d'un cycle donné.

4. Recopier et compléter sur votre copie le chronogramme ci-dessous pour les processus `p1`, `p2`, `p3` et `p4`.

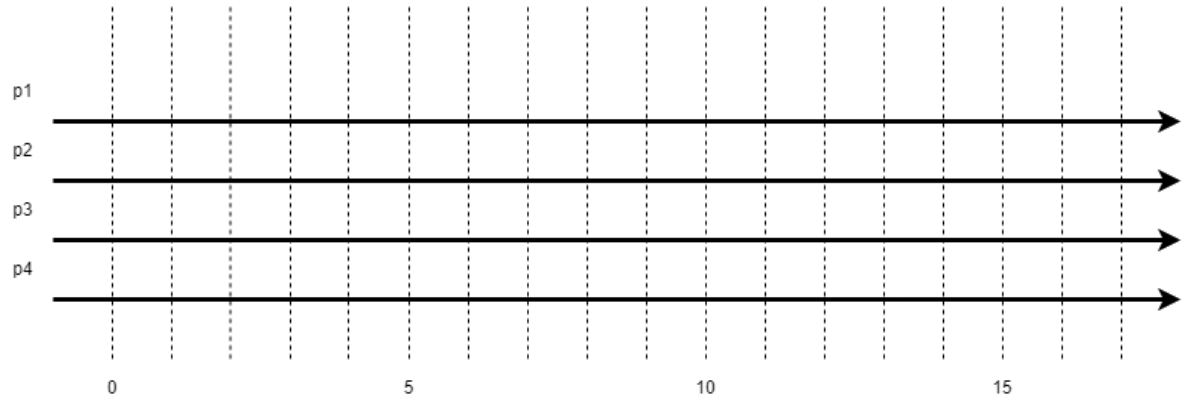


Figure 2. Chronogramme pour les processus p1, p2, p3 et p4

Pour mettre en place l'ordonnancement suivant cette méthode, on écrit la classe `Ordonnanceur` dont voici un code incomplet (l'attribut `temps` correspond au cycle en cours) :

```
1  class Ordonnanceur:
2
3      def __init__(self):
4          self.temps = 0
5          self.file = File()
6
7      def ajoute_nouveau_processus(self, proc):
8          '''Ajoute un nouveau processus dans la file de
9            l'ordonnanceur. '''
10          ...
11
12      def tourniquet(self):
13          '''Effectue une étape d'ordonnancement et renvoie le nom
14            du processus élu.'''
15          self.temps += 1
16          if not self.file.est_vide():
17              proc = ...
18              ...
19              if not proc.est_fini():
20                  ...
21              return proc.nom
22          else:
23              return None
```

5. Compléter le code ci-dessus.

À chaque appel de la méthode `tourniquet`, celle-ci renvoie soit le nom du processus qui a été élu, soit `None` si elle n'a pas trouvé de processus en cours.

6. Écrire un programme qui :

- utilise les variables `p1`, `p2`, `p3`, `p4` et `depart_proc` définies précédemment ;
- crée un ordonnanceur ;
- ajoute un nouveau processus à l'ordonnanceur lorsque c'est le moment ;
- affiche le processus choisi par l'ordonnanceur ;
- s'arrête lorsqu'il n'y a plus de processus à exécuter.

Dans la situation donnée en exemple (voir Figure 1), il s'avère qu'en fait les processus utilisent des ressources comme :

- un fichier commun aux processus ;
- le clavier de l'ordinateur ;
- le processeur graphique (GPU) ;
- le port 25000 de la connexion Internet.

Voici le détail de ce que fait chaque processus :

Liste des processus			
A	B	C	D
acquérir le GPU	acquérir le clavier	acquérir le port	acquérir le fichier
faire des calculs	acquérir le fichier	faire des calculs	faire des calculs
libérer le GPU	libérer le clavier	libérer le port	acquérir le clavier
	libérer le fichier		libérer le clavier
			libérer le fichier

7. Montrer que l'ordre d'exécution donné en exemple aboutit à une situation d'interblocage.