

## EXERCICE 2 (4 points)

*Cet exercice traite du thème « architecture matérielle », et principalement d'ordonnancement et d'expressions booléennes.*

Un système est composé de 4 périphériques, numérotés de 0 à 3, et d'une mémoire, reliés entre eux par un bus auquel est également connecté un dispositif ordonnanceur. À l'aide d'un signal spécifique envoyé sur le bus, l'ordonnanceur sollicite à tour de rôle les périphériques pour qu'ils indiquent le type d'opération (lecture ou écriture) qu'ils souhaitent effectuer, et l'adresse mémoire concernée.

Un tour a lieu quand les 4 périphériques ont été sollicités. **Au début d'un nouveau tour, on considère que toutes les adresses sont disponibles en lecture et écriture.**

Si un périphérique demande l'écriture à une adresse mémoire **à laquelle on n'a pas encore accédé** pendant le tour, l'ordonnanceur répond "OK" et l'écriture a lieu. Si on a déjà demandé la lecture ou l'écriture à cette adresse, l'ordonnanceur répond "ATT" et l'opération n'a pas lieu.

Si un périphérique demande la lecture à une adresse à laquelle on n'a pas encore accédé en écriture pendant le tour, l'ordonnanceur répond "OK" et la lecture a lieu. Plusieurs lectures peuvent avoir donc lieu pendant le même tour à la même adresse.

Si un périphérique demande la lecture à une adresse à laquelle on a déjà accédé en écriture, l'ordonnanceur répond "ATT" et la lecture n'a pas lieu.

Ainsi, pendant un tour, une adresse peut être utilisée soit une seule fois en écriture, soit autant de fois qu'on veut en lecture, soit pas utilisée.

Si un périphérique ne peut pas effectuer une opération à une adresse, il demande la même opération à la même adresse au tour suivant.

1. Le tableau donné en annexe 1 indique, sur chaque ligne, le périphérique sélectionné, l'adresse à laquelle il souhaite accéder et l'opération à effectuer sur cette adresse. Compléter dans la dernière colonne de cette annexe, à rendre avec la copie, la réponse donnée par l'ordonnanceur pour chaque opération.

On suppose dans toute la suite que :

- le périphérique 0 écrit systématiquement à l'adresse 10 ;
- le périphérique 1 lit systématiquement à l'adresse 10 ;
- le périphérique 2 écrit alternativement aux adresses 11 et 12 ;
- le périphérique 3 lit alternativement aux adresses 11 et 12 ;

Pour les périphériques 2 et 3, le changement d'adresse n'est effectif que lorsque l'opération est réalisée.

2. On suppose que les périphériques sont sélectionnés à chaque tour dans l'ordre 0 ; 1 ; 2 ; 3. Expliquer ce qu'il se passe pour le périphérique 1.

Les périphériques sont sollicités de la manière suivante lors de quatre tours successifs :

- au premier tour, ils sont sollicités dans l'ordre 0 ; 1 ; 2 ; 3 ;
- au deuxième tour, dans l'ordre 1 ; 2 ; 3 ; 0 ;
- au troisième tour, 2 ; 3 ; 0 ; 1 ;
- puis 3 ; 0 ; 1 ; 2 au dernier tour.
- Et on recommence...

3. a. Préciser pour chacun de ces tours si le périphérique 0 peut écrire et si le périphérique 1 peut lire.

b. En déduire la proportion des valeurs écrites par le périphérique 0 qui sont effectivement lues par le périphérique 1.

On change la méthode d'ordonnancement : on détermine l'ordre des périphériques au cours d'un tour à l'aide de deux listes d'attente ATT\_L et ATT\_E établies au tour précédent.

Au cours d'un tour, on place dans la liste ATT\_L toutes les opérations de lecture mises en attente, et dans la liste d'attente ATT\_E toutes les opérations d'écriture mises en attente.

Au début du tour suivant, on établit l'ordre d'interrogation des périphériques en procédant ainsi :

- on interroge ceux présents dans la liste ATT\_L, par ordre croissant d'adresse,
- on interroge ensuite ceux présents dans la liste ATT\_E, par ordre croissant d'adresse,
- puis on interroge les périphériques restants, par ordre croissant d'adresse.

4. Compléter et rendre avec la copie le tableau fourni en annexe 2, en utilisant l'ordonnancement décrit ci-dessus, sur 3 tours.

Les colonnes **e0** et **e1** du tableau suivant recensent les deux chiffres de l'écriture binaire de l'entier **n** de la première colonne.

nombre n	écriture binaire de n sur deux bits	e1	e0
0	00	0	0
1	01	0	1
2	10	1	0
3	11	1	1

L'ordonnanceur attribue à deux signaux sur le bus de données les valeurs de **e0** et **e1** associées au numéro du circuit qu'il veut sélectionner. On souhaite construire à l'aide des portes ET, OU et NON un circuit pour chaque périphérique.

Chacun des quatre circuits à construire prend en entrée deux signaux **e0** et **e1**, le signal de sortie **s** valant 1 uniquement lorsque les niveaux de **e0** et **e1** correspondent aux bits de l'écriture en binaire du numéro du périphérique correspondant.

Par exemple, le circuit ci-dessous réalise la sélection du périphérique 3. En effet, le signal **s** vaut 1 si et seulement si **e0** et **e1** valent tous les deux 1.



5. a. Recopier sur la copie et indiquer dans le circuit ci-dessous les entrées **e0** et **e1** de façon à ce que ce circuit sélectionne le périphérique 1.



- b. Dessiner un circuit constitué d'une porte ET et d'une porte NON, qui sélectionne le périphérique 2.

- c. Dessiner un circuit permettant de sélectionner le périphérique 0.

**Annexe 1 de l'exercice 2, à rendre avec la copie.**

Numéro du périphérique	Adresse	Opération	Réponse de l'ordonnanceur
0	10	écriture	OK
1	11	lecture	OK
2	10	lecture	ATT
3	10	écriture	ATT
0	12	lecture	
1	10	lecture	
2	10	lecture	
3	10	écriture	

**Annexe 2 de l'exercice 2, à rendre avec la copie**

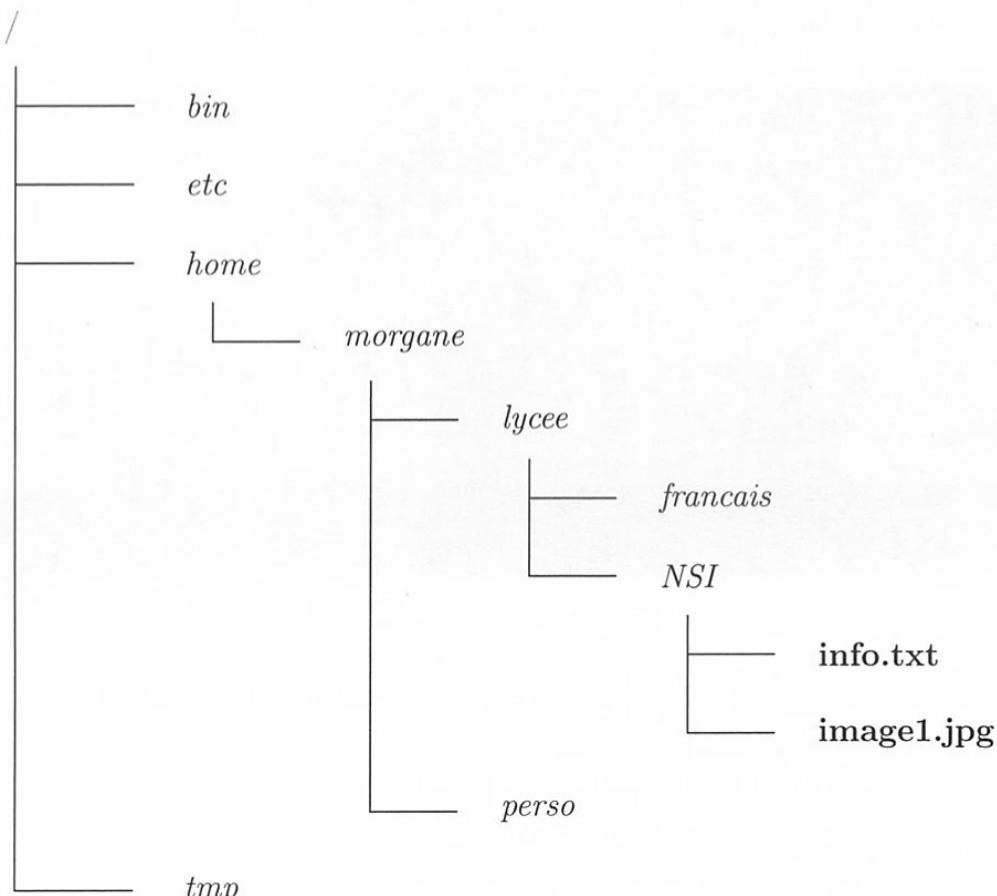
Tour	Numéro du périphérique	Adresse	Opération	Réponse de l'ordonnanceur	ATT_L	ATT_E
1	0	10	écriture	OK	vide	vide
1	1	10	lecture	ATT	(1,10)	vide
1	2	11	écriture			
1	3	11	lecture			
2	1	10	lecture			vide
2						
2						
3	0	10	écriture		vide	vide
3	1	10	lecture			vide
3	2	11	écriture	OK	(1,10)	vide
3	3	12	lecture			

## Exercice 2 (4 points).

Cet exercice porte sur les systèmes d'exploitation et la gestion des processus par un système d'exploitation.

Cet exercice pourra utiliser des commandes de systèmes d'exploitation de type UNIX telles que `cd`, `ls`, `mkdir`, `rm`, `rmdir`, `mv`, `cat`.

1. Dans un système d'exploitation de type UNIX, on considère l'arborescence des fichiers suivante dans laquelle les noms de dossiers sont en italique et ceux des fichiers sont en gras :



On souhaite, grâce à l'utilisation du terminal de commande, explorer et modifier les répertoires et fichiers présents.

On suppose qu'on se trouve actuellement à l'emplacement `/home/morgane`.

- Parmi les quatre propositions suivantes, donner celle correspondant à l'affichage obtenu lors de l'utilisation de la commande `ls`.

**Proposition 1** : lycee francais NSI info.txt image1.jpg perso

**Proposition 2** : lycee perso

**Proposition 3** : morgane

**Proposition 4** : bin etc home tmp

- Écrire la commande qui permet, à partir de cet emplacement, d'atteindre le répertoire `lycee`.

On suppose maintenant qu'on se trouve dans le répertoire `/home/morgane/lycee/NSI`.

- (c) Écrire la commande qui permet de créer à cet emplacement un répertoire nommé **algorithme**.
- (d) Écrire la commande qui permet, à partir de cet emplacement, de supprimer le fichier `image1.jpg`.
2. On rappelle qu'un processus est une instance d'application. Un processus peut être démarré par l'utilisateur, par un périphérique ou par un autre processus appelé parent.
- La commande UNIX `ps` présente un cliché instantané des processus en cours d'exécution.
- On a exécuté la commande `ps` (avec quelques options qu'il n'est pas nécessaire de connaître pour la réussite de cet exercice). Un extrait du résultat de la commande est présenté ci-dessous :

UID	PID	PPID	C	STIME	TTY	TIME	CMD
test	900	739	0	10:51	?	00:00:00	/usr/lib/gvfs/gvfs-udisks2-vol
test	907	838	0	10:51	?	00:00:00	/usr/lib/gvfs/gvfsd-trash --sp
test	913	739	0	10:51	?	00:00:00	/usr/lib/gvfs/gvfsd-metadata
test	918	823	0	10:51	?	00:00:00	/usr/lib/x86_64-linux-gnu/xfce
test	919	823	0	10:51	?	00:00:00	/usr/lib/x86_64-linux-gnu/xfce
test	923	1	0	10:51	?	00:00:02	xfce4-terminal
test	927	923	0	10:51	pts/0	00:00:00	bash
test	1036	1	0	11:18	?	00:00:02	mousepad /home/test/Documents/
test	1058	923	0	11:22	pts/1	00:00:00	bash
root	1132	2	0	11:37	?	00:00:00	[kworker/0:0-ata_sff]
root	1134	2	0	11:43	?	00:00:00	[kworker/0:2-ata_sff]
test	1140	739	0	11:43	?	00:00:00	/usr/lib/x86_64-linux-gnu/tumb
root	1149	2	0	11:43	?	00:00:00	[kworker/u2:0-events_unbound]
test	1153	927	0	11:44	pts/0	00:00:00	vi
test	1154	927	0	11:44	pts/0	00:00:00	python3 prog.py
test	1155	1058	0	11:44	pts/1	00:00:00	ps -aef

On rappelle que :

- l'UID est l'identifiant de l'utilisateur propriétaire du processus ;
  - le PID est l'identifiant du processus ;
  - le PPID est l'identifiant du processus parent ;
  - C indique l'utilisation processeur ;
  - STIME est l'heure de démarrage du processus ;
  - TTY est le nom du terminal de commande auquel le processus est attaché ;
  - TIME est la durée d'utilisation du processeur par le processus ;
  - CMD le nom de commande utilisé pour démarrer le processus.
- (a) Donner le PID du parent du processus démarré par la commande `vi`.
- (b) Donner le PID d'un processus enfant du processus démarré par la commande `xfce4-terminal`.
- (c) Citer le PID de deux processus qui ont le même parent.
- (d) Parmi tous les processus affichés, citer le PID des deux qui ont consommé le plus de temps du processeur.

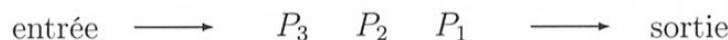
3. On considère les trois processus  $P_1$ ,  $P_2$  et  $P_3$ , tous soumis à l'instant 0 dans l'ordre 1, 2, 3 :

Nom du processus	Durée d'exécution en unité de temps	Ordre de soumission
$P_1$	3	1
$P_2$	1	2
$P_3$	4	3

- (a) Dans cette question, on considère que les processus sont exécutés de manière concurrente selon la politique du tourniquet : le temps est découpé en tranches nommées *quantums de temps*.

Les processus prêts à être exécutés sont placés dans une file d'attente selon leur ordre de soumission.

Lorsqu'un processus est élu, il s'exécute au plus durant un quantum de temps. Si le processus n'a pas terminé son exécution à l'issue du quantum de temps, il réintègre la file des processus prêts (côté entrée). Un autre processus, désormais en tête de la file (côté sortie) des processus prêts, est alors à son tour élu pour une durée égale à un quantum de temps maximum.



Reproduire le tableau ci-dessous sur la copie et indiquer dans chacune des cases le processus exécuté à chaque cycle. Le quantum correspond à une unité de temps.

<b>P1</b>							
0	1	2	3	4	5	6	7

- (b) Dans cette question, on considère que les processus sont exécutés en appliquant la politique du « plus court d'abord » : les processus sont exécutés complètement dans l'ordre croissant de leurs temps d'exécution, le plus court étant exécuté en premier.

Reproduire le tableau ci-dessous sur la copie et indiquer dans chacune des cases le processus exécuté à chaque cycle.

0	1	2	3	4	5	6	7

4. On considère trois ressources  $R_1$ ,  $R_2$  et  $R_3$  et trois processus  $P_1$ ,  $P_2$  et  $P_3$  dont les files d'exécution des instructions élémentaires sont indiquées ci-dessous :

Processus $P_1$	Processus $P_2$	Processus $P_3$
Demande $R_1$	Demande $R_2$	Demande $R_3$
Demande $R_2$	Demande $R_3$	Demande $R_1$
Libère $R_1$	Libère $R_2$	Libère $R_3$
Libère $R_2$	Libère $R_3$	Libère $R_1$

- (a) Rappeler les différents états d'un processus et expliquer pourquoi il y a ici risque d'interblocage, en proposant un ordre d'exécution des instructions élémentaires le provoquant.
- (b) Proposer un ordre d'exécution des instructions élémentaires sans interblocage.

4. Écrire une fonction `supprime_toutes_occurrences(pile, element)` qui prend en arguments un objet Pile appelé `pile` et un élément `element` et supprime tous les éléments `element` de `pile`.

On donne ci-dessous les résultats attendus pour certaines instructions.

```
>>>pile.afficher()  
7, 5, 2, 3, 5  
>>>supprime_toutes_occurrences (pile, 5)  
>>>pile.afficher()  
7, 2, 3
```

## Exercice 3 (4 points)

Cet exercice porte sur la gestion des processus par un système d'exploitation et les protocoles de routage.

**Les parties A et B sont indépendantes.**

### Partie A : Processus

La commande UNIX `ps` présente un cliché instantané des processus en cours d'exécution.

Avec l'option `-eo pid,ppid,stat,command`, cette commande affiche dans l'ordre l'identifiant du processus PID (*process identifier*), le PPID (*parent process identifier*), l'état STAT et le nom de la commande à l'origine du processus.

Les valeurs du champ STAT indique l'état des processus :

R : processus en cours d'exécution

S : processus endormi

Sur un ordinateur, on exécute la commande `ps -eo pid,ppid,stat,command` et on obtient un affichage dont on donne ci-dessous un extrait.

```
$ ps -eo pid,ppid,stat,command

  PID  PPID STAT COMMAND
    1      0   Ss   /sbin/init
    ...   ...
 1912  1908  Ss   Bash
 2014  1912  Ss   Bash
 1920  1747  S1   Gedit
 2013  1912  Ss   Bash
 2091  1593  S1   /usr/lib/firefox/firefox
 5437  1912  S1   python programme1.py
 5440  2013  R    python programme2.py
 5450  1912  R+   ps -eo pid,ppid,stat,command
```

À l'aide de cet affichage, répondre aux questions ci-dessous.

1. Quel est le nom de la première commande exécutée par le système d'exploitation lors du démarrage ?
2. Quels sont les identifiants des processus actifs sur cet ordinateur au moment de l'appel de la commande `ps` ? Justifier la réponse.
3. Depuis quelle application a-t-on exécuté la commande `ps` ?  
Donner les autres commandes qui ont été exécutées à partir de cette application.
4. Expliquer l'ordre dans lequel les deux commandes `python programme1.py` et `python programme2.py` ont été exécutées.
5. Peut-on prédire que l'une des deux commandes `python programme1.py` et `python programme2.py` finira avant l'autre ?

## Exercice 4 (4 points)

Cet exercice porte sur la gestion des processus et des ressources par un système d'exploitation.

Les parties A et B peuvent être traitées indépendamment.

### A. Ordonnancement des processus

Dans le laboratoire d'analyse médicale d'un hôpital, plusieurs processus peuvent demander l'allocation du processeur simultanément.

Le tableau ci-dessous donne les demandes d'exécution de quatre processus et indique :

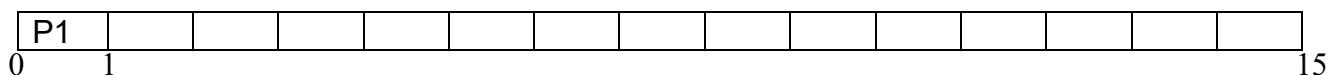
- le temps d'exécution du processus (en unité de temps) ;
- l'instant d'arrivée du processus sur le processeur (en unité de temps) ;
- le numéro de priorité du processus (classé de 1 à 10).

Plus la priorité est grande plus le numéro de priorité est petit.

Ainsi le processus P3, du tableau ci-dessous, est plus prioritaire que le processus P1. L'ordonnancement est de type préemptif, ce qui signifie qu'à chaque unité de temps, le processeur choisit d'exécuter le processus ayant le plus petit numéro de priorité (un seul processus à la fois). Ceci peut provoquer la suspension d'un autre processus qui reprendra lorsqu'il deviendra le plus prioritaire dans la file d'attente.

Processus	Temps d'exécution	Instant d'arrivée	Numéro de priorité
P1	3	0	4
P2	4	2	2
P3	3	3	1
P4	4	5	3

1. Reproduire le diagramme ci-dessous, sur votre copie, et indiquer dans chacune des cases le processus exécuté par le processeur entre deux unités de temps (il peut y avoir des cases vides).



2. Recopier et compléter les temps de séjour ainsi que les temps d'attente de chacun des processus (toujours en unités de temps).

$$\text{Temps de séjour} = \text{instant de terminaison} - \text{instant d'arrivée}$$

$$\text{Temps d'attente} = \text{temps de séjour} - \text{temps d'exécution}$$

Processus	Temps d'exécution	Instant d'arrivée	Temps de séjour	Temps d'attente
P1	3	0	$14 - 0 = 14$	$14 - 3 = 11$
P2	4	2		
P3	3	3		
P4	4	5		

3. À quelles conditions le temps d'attente d'un processus peut-il être nul ?

## B. Processus et ressources

Dans ce laboratoire d'analyse médicale de l'hôpital, le laborantin en charge du traitement des différents prélèvements (sanguins, urinaires et biopsiques) utilise simultanément quatre logiciels :

- Logiciel d'analyse d'échantillons (connecté à l'analyseur)
- Logiciel d'accès à la base de données des patients (SGBD)
- Traitement de texte
- Tableur

Le tableau ci-dessous donne l'état à un instant donné des différents processus (instances des programmes) qui peuvent soit mobiliser (M) des données (D1, D2, D3, D4 et D5), soit être en attente des données (A) ou ne pas les solliciter (-).

Une donnée ne peut être mobilisée que par un seul processus à la fois. Si un autre processus demande une donnée déjà mobilisée, il passe en attente.

Exemple : le SGBD mobilise la donnée D4 et est en attente de la donnée D5

	D1	D2	D3	D4	D5
Analyseur échantillon	M	-	-	A	-
SGBD	-	-	-	M	A
Traitement de texte	-	M	A		-
Tableur	A	-	M	-	M

1. À partir du tableau ci-dessus, démontrer que, à cet instant, les processus s'attendent mutuellement.
2. Comment s'appelle cette situation ?
3. On suppose que l'analyseur d'échantillon libère la ressource D1. Donner un ordre possible d'exécution des processus.