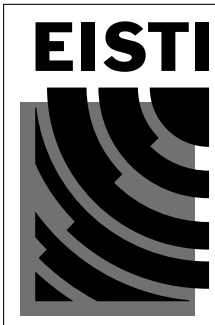	**EX - Examen n°1 (corrigé)**
T. Gherbi - J.A. Lorenzo	Système d'exploitation
ING1-GI	Année 2017–2018

Modalités

- Durée : 2 heures.
- Vous devez rédiger votre copie à l'aide d'un stylo à encre exclusivement.
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Une seule feuille manuscrite (pas de photocopies) est autorisée.
- Aucune question ne peut être posée aux enseignants, posez des hypothèses en cas de doute.
- Aucune machine électronique ne doit se trouver sur vous ou à proximité, même éteinte.
- Aucune sortie n'est autorisée avant une durée incompressible d'une heure.
- Aucun déplacement n'est autorisé.
- Aucun échange, de quelque nature que ce soit, n'est possible.

Exercice 1 : Programmation de processus (3 points)

Écrivez un programme C qui crée un fils. Chaque processus doit afficher son PID à l'écran. Ensuite, le père doit attendre la terminaison du fils. Lorsque le fils termine, il enverra un code de retour, qui doit être récupéré par le père et affiché à l'ecran.

Exercice 2 : Gestion des fichiers (4 points)

Supposons un noeud d'information (*i-node*) en Unix (10 adresses de bloc directs et 3 indirects) contenant un fichier. Quelle est la taille maximale d'un fichier si nous avons des blocs de 2 Kbytes et le numéro de bloc est donné sur 16 bits ? (**Note :** Vous n'avez pas de calculatrice, donc il suffit de proposer la réponse même si vous ne faites pas toutes les opérations).

2 KB / 2 bytes (16 bits) = 1024 numéros de blocs dans un bloc.

blocs directs = 10 blocs

bloc indirect_1 : 1024 blocs

bloc indirect_2 : 1024^2 blocs

bloc indirect_3 : 1024^3 blocs

Nombre maximum de blocs dans un fichier : $10 + 1024 + 1024^2 + 1024^3 = 1074791434$

Taille maximale du fichier : 1074791434 blocs * 2 KB/bloc = 2149582868 Kbytes ~ 2 TBytes.

Note : les élèves n'ont pas de calculatrice, donc il suffit de proposer la réponse même s'ils ne font pas toutes les opérations.

Exercice 3 : Ordonnanceur (4 points)

Soient trois processus A, B et C prêts tels que A est arrivé en premier suivi de B, 2 unités de temps après et C, 1 unité de temps après B. Les temps nécessaires pour l'exécution des processus A, B et C sont respectivement 8, 4 et 2 unités de temps. Le temps de commutation est supposé nul. Calculer :

- le **temps de séjour** de chaque processus.
- le **temps moyen de séjour**.
- le **temps d'attente** : temps de séjour - temps d'exécution du travail.
- le **temps moyen d'attente**.
- le nombre de **changements de contexte** (Note : un changement de contexte se produit à chaque fois qu'un processus acquiert le processeur)

en utilisant les techniques :

1. SRT (Shortest Remaining Time)
2. Round robin (quantum = 3 unités de temps)

Processus	**Temps d'exécution**	**Temps d'arrivage**
A	8	0
B	4	2
C	2	3

Réponse :

1. SRT :

1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	A	B	C	C	B	B	B	A	A	A	A	A	A

B arrive à $t = 2$ et, comme il est plus court que le temps qui reste d'A, B est mis en exécution directement. C arrive à $t = 3$ et, comme il est plus court que le temps qui reste de B, C est mis en exécution à la place de B.

Processus	**Temps de séjour**	**Temps d'attente**
A	14-0=14	14-8=6
B	8-2=6	6-4=2
c	5-3=2	2-2=0

Temps moyen de séjour = $\frac{14+6+2}{3} = \frac{22}{3} = 7,33$

Temps moyen d'attente = $\frac{6+2+0}{3} = 2,66$

5 changement de contexte (en comptant le changement initial pour l'arrivage d'A)

3. RR (Q = 3) :

1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	A	A	B	B	B	C	C	A	A	A	B	A	A

Après l'unité de temps 3, on change à B pendant 3 unités de temps. C est déjà arrivé et mis en queue donc il s'exécute pendant les 2 unités de temps. Une fois que C termine, on change à B et

après à A jusqu'à la fin.

Processus	Temps de séjour	Temps d'attente
A	14-0=14	14-8=6
B	12-2=10	10-4=6
C	8-3=5	5-2=3

Temps moyen de séjour = $\frac{14+10+5}{3} = 9,7$

Temps moyen d'attente = $\frac{6+6+3}{3} = 5$

6 changements de contexte (en comptant le changement initial pour l'arrivage d'A)

Exercice 4 : Conteneurs (3 points)

Supposons que je démarre un conteneur Docker avec la commande `docker run -it ubuntu /bin/bash`. Ensuite je modifie un fichier texte à l'intérieur du conteneur et, finalement, je sors du conteneur avec `exit`. Plus tard, je redémarre le conteneur avec la même commande. Le fichier texte contiendra-t-il les modifications apportées ? Si oui, justifiez. Si non, expliquez pourquoi et dites s'il est possible de retrouver la version modifiée du fichier texte.

La deuxième fois qu'on exécute la commande, ce qu'on fait est démarrer un nouveau conteneur, donc il ne contiendra pas le fichier modifié. Pour récupérer le fichier modifié il faudra chercher le conteneur qui est arrêté (avec `docker ps -a`) et le redémarrer (avec `docker start` et `docker attach`).

Questions courtes (6 points)

(1) Dans le code suivant, combien de processus sont crées lorsque le programme est exécuté ? □

```
int main() {
        fork();
        fork();
        exit();
}
```

Quatre.

(2) Quelles sont les avantages et les inconvenants d'un système d'allocation chaînée dans un disque dur ? ☐

Avantages : facilite les modifications et l'accroissement.
Inconvénients : alourdit la recherche d?une donnée. Pas d?accès aléatoire mais séquentiel

(3) Dans quelle région en mémoire sont placées les données allouées avec `malloc()` ? ☐

Dans le Heap ou Tas.

(4) Quelle est la différence entre un ordonnanceur sans réquisition (OSR) et avec réquisition (OAR) ? ☐

(5) Un programme contient une variable *a*. Après avoir appelé `fork()`, le fils contiendra une copie de la même variable, qui pourra être modifié indépendemment de celle du père. Si on vérifie l'adresse de cette variable dans le père et dans le fils, nous trouvons qu'elle est la même. Comment peut-on avoir deux variables avec la même adresse en mémoire mais avec des valeurs différentes ? ☐

Cela a été vu dans le TP1 et expliqué à nouveau dans la séance de mémoire : Elles sont deux variables différentes. Les deux processus vont voir la même `adresse virtuelle` (conséquence du `fork()`) mais l'adresse physique sera différente pour chaque variable.

(6) C'est quoi un *défaut de page* et à quoi sert le *bit V* (valid-bit) de la table de pages ? ☐