CY TECH	**Système d'exploitation** **Examen Session Rattrapage**
I. Ben Amor, S. Bouzidi, T. Gherbi, J.A. Lorenzo, S.Yassa.	
ING1-GI-GM	Année 2020–2021

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.
- Aucun document n'est autorisé.
- La calculatrice est autorisée.
- Aucune question ne peut être posée aux enseignants, posez des hypothèses en cas de doute.
- 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.

QCM à répondre sur votre copie et non pas sur ce sujet (5 points)

a) Le Système d'Exploitation...

- ❏ est une couche matérielle qui gère le logiciel de présentation.
- ❏ est responsable du partage des ressources matérielles de l'ordinateur.
- ❏ est stocké dans le secteur 0 du MBR.
- ❏ Aucune des réponses n'est valide.

b) Un appel système...

- ❏ constitue une interface entre l'interface graphique et la ligne de commande.
- ❏ s'exécute au niveau utilisateur.
- ❏ entre dans le kernel.
- ❏ Aucune des réponses n'est valide.

c) La commande *parted*...

- ❏ est un outil de partitionnement.
- ❏ permet d'étendre la taille du système de fichiers d'une partition.
- ❏ permet d'entrer dans le mode de récupération du système d'exploitation.
- ❏ Aucune des réponses n'est valide.

d) Un tableau alloué avec malloc() sera stocké dans

- ❏ la région du code (text segment).
- ❏ la région des données (data segment).
- ❏ la pile (stack).
- ❏ le tas (heap).

e) Pour créer une image à partir d'un conteneur Docker en exécution nous utiliserons

- ❏ docker build
- ❏ docker ps
- ❏ docker commit
- ❏ docker-compose

Questions de cours (5 points)

a. Décrire le processus de démarrage de Linux.
b. Expliquer la différence entre l'allocation contiguë et l'allocation chaînée de fichiers, en précisant les avantages et inconvénients de chacune.
c. Que contient le MBR (Master Boot Record) ?
d. Expliquez le rôle de la zone de swap.
e. Expliquez comment un processus peut devenir zombie.

Exercice 1 : Programmation de processus (4 (2+2) points)

1- Combien de processus sont créés par le programme suivant ? Expliquez.

```
# include <unistd.h>
# include <stdio.h>

int main() {
      int pid, i;
      for (i=0; i<2; i++)
      {
            pid = fork();
            if (pid < 0)
            {
                  printf("echec de fork\n");
                  exit(-1);
            }
      }
      return 0 ;
}
```

2- Ecrire un programme qui crée un processus fils. Ce dernier envoie à son père la valeur 20. Le père utilise waitpid pour attendre son fils, récupère la valeur envoyée par le fils et l'affiche.

Exercice 2 : Le système de fichiers (2 points)

Considérons un système de fichiers de type Unix dans lequel l'information concernant les blocs de données de chaque fichier est accessible à partir du i-noeud.

Supposons que :

- Le système de fichiers utilise des blocs de données de taille fixe 2K (2048 bytes).
- L'i-noeud de chaque fichier (ou répertoire) contient 12 pointeurs directs sur des blocs de données, 1 pointeur indirect simple, 1 pointeur indirect double et 1 pointeur indirect triple.
- Chaque pointeur (numéro de bloc) est représenté sur 4 bytes.

a) Quelle est la plus grande taille de fichier que peut supporter ce système de fichiers?

b) Considérons un fichier de taille 1 MByte (1024 KB). Combien de blocs de données pointés **par les blocs d'indirection** sont-ils nécessaires pour représenter ce fichier sur le disque ?

Exercice 3 : Gestion de processus - Ordonnancement (2 points)

Soient quatre processus prêts A, B, C et D avec les temps d'arrivée au système et les temps d'exécution indiqués dans le tableau ci-dessous :

Processus	Temps d'exécution	Temps d'arrivée
A	3	0
B	3	1
C	2	3
D	2	7

En supposant que le temps de commutation est nul, calculer :

— le temps de séjour de chaque processus.

— le temps moyen de séjour.

— le temps d'attente : temps de séjour - temps d'exécution.

— le temps moyen d'attente.

— le nombre de changements de contexte

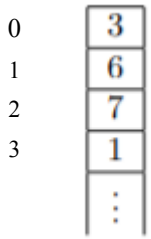
en utilisant les techniques :

1. *First-come, first-served (FCFS).*
2. *Round-robin* avec un quantum de 2.

Pour chaque technique, dessiner la séquence des processus qui s'exécutent à chaque moment.

Exercice 4 : Mémoire virtuelle (2 points)

Dans une architecture hypothétique où les adresses virtuelles sont sur 14 bits, les 4 bits à gauche dans l'adresse dénotent la page et les 10 bits à droite dénotent le déplacement par rapport au début de la page (offset). En sachant que les premières entrées de la table des pages (correspondance « pages virtuelles – cadres réelles ») sont :



Transformez les deux adresses virtuelles suivantes (exprimées en notation binaire), en adresses physiques qui sont sur 13 bits.

- 101010101010
- 1

Corrigé de l'examen

QCM : (5 points)

- Le Système d'Exploitation...

  ❑ est responsable du partage des ressources matériels de l'ordinateur.

- Un appel système...

  ❑ entre dans le kernel.

- La commande *parted*...

  ❑ est un outil de partitionnement.

- Un tableau alloué avec malloc sera stocké dans

  ❑ le tas (heap).

- Pour créer une image à partir d'un conteneur Docker en exécution nous utiliserons

  ❑ docker commit

Questions de cours : (5 points)

- Décrire le processus de démarrage de Linux (chap3, page 8)
- Expliquer la différence entre l'allocation contiguë et l'allocation chaînée de fichiers, en précisant les avantages et inconvénients de chacune. (réponse dans chap2, pages 17, 18)
- Que contient le MBR (Master Boot Record)? (réponse en chap 2, page 24)
- Expliquez le rôle de la zone de swap?
- Expliquez comment un processus peut devenir zombie?

Exercice 1 : Programmation de processus (4 points)

1- Combien de processus sont créés par le programme suivant ? Expliquez. réponse 4

2- Ecrire un programme qui crée un processus fils. Ce dernier envoie à son père la valeur 20. Le père utilise waitpid pour attendre son fils, récupère la valeur envoyée par le fils et l'affiche.

```
# include <unistd.h>
# include <stdio.h>

int main() {
        int pid, status;
        pid = fork();
        if (pid < 0)
        {
                printf("echec de fork\n");
                exit(-1);
```

```
        } else if (pid > 0) {  // partie du père
                        waitpid(pid, &status);
                        printf ("Valeur envoyee par l'enfant : %d\n", WEXITSTATUS(status));
        } else { // partie enfant
                        exit(20);
                }
        }
        return 0 ;
}
```

Exercice 2 : Le système de fichiers (2 points)

a) Chaque pointeur (numéro de bloc) est représenté sur 4 bytes. Donc chaque bloc de données peut contenir 512 pointeurs (car 2K/4)

(12 * 2k) pour direct, (512 * 2K) pour indirect simple, (512^2 * 2K) pour indirect double,

(256^3 * 2K) pour indirect Triple.

La plus grande taille de fichier est :

Taille maximale = (12 + 512 + 512^2 + 512^3) * 2K = 268960792 KBytes ~ 256,5 Go

b) Pour un fichier contenant 1 MB (1024 KB), on remarque que :

1024 KB = 512 * 2 KB

Il faudra donc 512 blocs pour conserver les données de ce fichier.

L'i-noeud ne dispose que de 12 blocs directs (512 - 12 = 500), il va donc falloir utiliser les pointeurs indirects pour conserver le reste des données. Nous aurons besoin des i-noeuds du pointeur indirect simple. En effet, il nous restera 500 blocs qui seront tous stockés sur la partie pointée par le pointer d'indirection simple, car 500 est inférieur à 512. Donc la réponse est **500 blocs**.

Exercice 3 : Gestion de processus - ordonnancement (2 points)

1. FCFS

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A	A	A	B	B	B	C	C	D	D									

A arrive B arrive C arrive D arrive

Queue :

t = 0 -> A (dans CPU)
t = 1 -> B, A (dans CPU)
t = 2 -> B, A (dans CPU)
t = 3 -> C, B (dans CPU)
t =4 -> C, B (dans CPU)
t =5 -> C, B (dans CPU)
t =6 -> C (dans CPU)
t =7 -> D, C (dans CPU)
t =8 -> D (dans CPU)

Processus	Temps séjour	Temps d'attente
A	3 - 0 = 3	3 - 3 = 0
B	6 - 1 = 5	5 - 3 = 2
C	8 - 3 = 5	5 - 2 = 3
D	10 - 7 = 3	3 - 2 = 1

T_moyen_sejour = (3+5+5+3) / 4 = 4 unités de temps
T_moyen_attente = (0+2+3+1) / 4 = 1,5 unités de temps
4 changements de contexte (en comptant le changement initial)

2. RR (Q=2)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A	A	B	B	A	C	C	B	D	D									

A arrive B arrive C arrive D arrive

Queue :
t = 0 -> A (dans CPU)
t = 1 -> B, A (dans CPU)
t = 2 -> A, B (dans CPU)
t = 3 -> C, A, B (dans CPU)
t = 4 -> B, C, A (dans CPU)
t = 5 -> B, C (dans CPU)
t = 6 -> B, C (dans CPU)
t = 7 -> D, B (dans CPU)
t = 8 -> D (dans CPU)

Processus	Temps séjour	Temps d'attente
A	5 - 0 = 5	5 - 3 = 2
B	8 - 1 = 7	7 - 3 = 4
C	7 - 3 = 4	4 - 2 = 2
D	10 - 7 = 3	3 - 2 = 1

T_moyen_sejour = (5+7+4+3) / 4 = 4,75 unités de temps
T_moyen_attente = (2+4+2+1) / 4 =2,25 unités de temps
6 changements de contexte (en comptant le changement initial)

Exercice 4 : Mémoire virtuelle (2 points)

- 0010 1010101010 : la page demandée est 0010 en binaire → donc 2, le déplacement étant 1010101010.

  Le cadre correspondant à 2 est 7, ce qui donne 111 en binaire.

  L'adresse physique correspondant est 111 1010101010

- 0000 0000000001 : la page est 0. Le cadre correspondant étant 3, ce qui donne 011 en binaire

  L'adresse physique correspondant est 011 0000000001.