Correction de l'Examen Systèmes d'exploitation.

Exercice 1 : (5 points)

1. Les états d'un processus :
   - **Prêt** : Le processus est prêt à s'exécuter mais attend dans une file d'attente des processus prêt que l'ordonnanceur lui alloue le microprocesseur. **(0.5)**
   - **En cours d'exécution** : Le processus a été élu par l'ordonnanceur pour s'exécuter dans le microprocesseur. Dans un ordinateur monoprocesseur, un seul processus au maximum est dans cet état **(0.5)**
   - **Bloqué** : Le processus est en attente dans la file d'attente des processus bloqués la fin d'une opération d'Entrée/Sortie. (ex : Accès disque). **(0.5)**

   Et les transitions :
   - **1** : Un processus en cours d'exécution fait une opération d'Entré/Sortie. **(0.25)**
   - **2** : Un processus en cours d'exécution est interrompu par l'ordonnanceur. pour permettre à un autre processus de s'exécuter sur le microprocesseur. **(0.25)**
   - **3** : Un processus prêt a été élue par l'ordonnanceur pour s'exécuter sur le microprocesseur. **(0.25)**
   - **4** : La fin d'une opération d'Entrée/Sortie lancé par un processus. **(0.25)**

2. Chaque processus dispose de ses propres ressources (Mémoire, fichiers, …) par contre un plusieurs threads peuvent partager les mêmes ressources. Les processus est threads se différencient également dans les aspects suivants : **(1)**

   - **Sécurité** : Les processus sont plus sûrs et plus robuste que les threads. Si un processus est corrompu, il n'impacte pas les autres. Un thread corrompu impacte tous les autres threads créés dans le même processus. **(0.25)**
   - **Performanc**e : La création d'un thread est plus rapide que la création d'un processus. **(0.25)**

3. Une implémentation des thread dans l'espace noyau est gérée par le système d'exploitation (ex : appel système Linux *clone()*). Par contre, une implémentation des threads dans l'espace utilisateur est gérée par une bibliothèque dans le système d'exploitation (ex : La bibliothèque Linux *libpthread*). **(1)**

Exercice 2 : (5 points)

1. **(1)**

Processus	Temps d'arrivée	Temps CPU	FCFS	SJF sans Préemption	SJF avec Préemption
P0	0	7	0	0	0
P1	3	3	12	5	0
P2	4	1	14	3	0
P3	1	8	6	10	10
P4	13	4	6	6	0
Moyenne			7.6	4.8	2

2. L'inconvénient de FCFS : Un long processus utilisant beaucoup de temps CPU pénalise les processus utilisant peu de temps CPU et effectuant beaucoup d'E/S. **(0.5)**

3. Voir tableau plus haut. **2.5 (1.5 avec préemption et 1 sans préemption)**
4. Dans l'algorithme d'ordonnancement *Round-Robin* (Tourniquet) :

- Un quantum trop petit ➔ Le système d'exploitation passe beaucoup de temps dans la commutation de contexte ➔dégradation des performances **(0.5)**
- Un quantum trop grand ➔ L'ordonnancement tend vers l'algorithme FCFS. **(0.5)**

Exercice 3 : (5 points)

Le problème posé s'apparente à un problème de « Rendez-vous »

```
Etat_train=’A’ ;
Nb_Portes_Ouvertes=0;
Semaphore mutex=1 ;
Semaphore sync=0;
```

(1)

***Démarrer_train*() (1)**

```
{
P(mutex)
If (Nb_Portes_Ouvertes > 0) {        // Il existe au moins un porte ouverte
        P(sync)                      // Se bloquer
        }
Démarrer …
Etat_train=’M’
V(mutex)
}
```

***Arrêter_train*() (1)**

```
{
P(mutex)
Arrêter …
Etat_train=’A’ ;
V(mutex)
}
```

***Ouvrir_porte*()(1)**

```
{
P(Mutex)
If (Etat_train==’A’)  {              // Ouvrir une porte seulement si le train est en arrêt
        Nb_Portes_Ouvertes= Nb_Portes_Ouvertes+1 ;
        }
V(Mutex)
}
```

***Fermer_auto_porte*()(1)**

```
{
P(Mutex)
Nb_Porte_Ouvertes= Nb_Porte_Ouvertes-1 ;
If (Nb_Porte_Ouvertes == 0) {        // La dernière porte a été fermée
        V(sync)                      // Débloquer le démarrage  du train
        }
V(Mutex)
}
```

Exercice 4 : **(5 points)**

1. Structure d'une adresse virtuelle :

   Taille de la page 8 KO = 2^{13} Octets ➔ Nombre de bits pour le champ déplacement = 13 (**0.5**)

   Mémoire virtuelle = 32 GO = 2^{35} ➔ Nombre de page = $\frac{2^{35}}{2^{13}} = 2^{22}$ ➔ Nombre de bits pour le champ numéro de page = 22 (**0.5**.)

   Adresse virtuelle :

Numéro de page (22 bits)	Déplacement (13 bits)

   Taille de la mémoire Physique = 256 MO = 2^{28} ➔ Nombre de frame = $\frac{2^{28}}{2^{13}} = 2^{15}$ ➔ Taille du champ numéro de la frame = 15 bits.

   Taille d'une entrée de la table de page = Numéro de frame + P + R + M = 18 bits = 3 Octets. (**0.5**)

2. Adresse Physique = (Numéro de la frame) * 2^{13} + déplacement. **0.5**