TD1 SE

EXERCICE1 : Aspects Genereaux des Processus

1.)La commande ps-aux permet de lister tous les processus du système.

2.a)la signifcation de chaque colonne **:**

User (Utilisateur): Le nom de l'utilisateur propriétaire du processus.

1. **PID (Process ID):** L'identifiant unique du processus.
2. **%CPU:** Le pourcentage de temps CPU utilisé par le processus depuis son démarrage.
3. **%MEM:** Le pourcentage de la mémoire physique utilisée par le processus.
4. **VSZ (Virtual Memory Size):** La taille totale de la mémoire virtuelle du processus en kilo-octets (KB).
5. **RSS (Resident Set Size):** La taille de la mémoire résidente utilisée par le processus en kilo-octets (KB). Il s'agit de la portion de la mémoire virtuelle actuellement présente dans la RAM.
6. **TTY (Terminal Type):** Le type de terminal associé au processus. Indique où le processus est actuellement en cours d'exécution.
7. **STAT (Process Status):** Le statut du processus, indiquant son état actuel (par exemple, S pour "Sleeping", R pour "Running", etc.).
8. **START (Start Time):** L'heure de démarrage du processus.
9. **TIME:** Le temps total CPU que le processus a consommé depuis son démarrage.
10. **COMMAND:** La commande ou le nom du programme associé au processus.

2.b) le système a démarré a 11:15

2.c) l ’utilité de UDISKS

`udisks` facilite la gestion des périphériques de stockage amovibles sur les systèmes Unix/Linux, en offrant une interface uniforme et en simplifiant les opérations courantes associées aux disques externes, clés USB, et autres dispositifs de stockage amovibles.

2.d)À partir de quelle heure jse a commencé à utiliser l’ordinateur ?

À partir de ces informations, on peut déduire que l'utilisateur `jse` a commencé à utiliser l'ordinateur à partir de 11:18.

2.e)que fait l ’utilisateur jse au moment du listening des processus ?

Au moment du listening jse a exécuté la commande pour regarder un film (`/usr/bin/playfilm.mp4`) à partir de 11:19. La commande `playfilm.mp4` est lancée en arrière-plan (`&` à la fin de la ligne), ce qui signifie que l'utilisateur peut continuer à utiliser le terminal tout en regardant le film.

2.f)Proposer une arborescence des processus correspondante à l’état du système listé en 1.2 ?

Init

|

| udisks

|Kde4

graphicalui

soundunit

konsole

play

soundserver

ffdcodec

ps -aux

2.g)Que se passe-t-il si l’utilisateur jse tape la commande Kill -9 1680 ?

La commande `kill -9 1680` envoie un signal d'interruption forcée (SIGKILL) à un processus en cours. Si elle est exécutée en tant qu'utilisateur non root sur un processus root, elle affichera un message "permission non accordée" car seuls les utilisateurs root ont le privilège de tuer les processus root.

2.h)Que se passe-t-il si l’utilisateur root tape la commande Kill -15 1720 ?

La commande `kill -15 1720` envoie un signal SIGTERM au processus, ordonnant une terminaison ordonnée. Lorsque exécutée par l'utilisateur root sur le processus playfilm, celui-ci sera interrompu, entraînant l'arrêt du film de manière contrôlée.

2.i)Quelle est la quantité de RAM en ko utiliser sur la machine par l’utilisateurs jse ?

11012 + 36624 + 16421 + 25332) / 1024 = 87.29 ko

jse a utilisé 87.29 ko .

Exercice 2 : Ordonnancement sans E/S

Partie 1 : 5 processus, P1, P2, P3, P4, P5 sont dans une file d’attente dans cet ordre (P1 est le premier, P5 est le dernier). Leur exécution demande un temps total de service exprimé en unités arbitraires :

Processus	P1	P2	P3	P4	P5
Temps de service estimé	10	1	2	1	5

1.Décrire l’exécution des processus dans le cadre des politiques d’ordonnancement FIFO, SJF, RR (avec un quantum de 1).

Pour FIFO :

Avec FIFO, le processeur est donné au premier processus arriver. Donc P1 ensuite P2,…, P5 .

On applique le principe du premier arrivé premier servit

P5	P4	P3	P2	P1 **CPU**

Pour SJF
L'algorithme SJF (Shortest Job First) organise les processus en fonction de leur temps d'exécution estimé le plus court, plaçant ainsi en tête de la file d'attente le processus qui nécessite le moins de temps pour s'exécuter. Dans le cas où deux processus ont la même durée d'exécution estimée (comme P2 et P4), l'algorithme peut faire un choix arbitraire entre eux. Par la suite, les processus P3, P5 et P1 sont placés dans la file d'attente dans l'ordre. Cela favorise l'exécution rapide des processus, réduisant le temps total d'exécution du lot de processus.

P1	P5	P3	P2/P4	**CPU**

Pour RR

Avec l'algorithme RR (Round Robin) et un quantum de 1, le système effectuera une rotation de processus à chaque unité de temps. Chaque processus sera exécuté pendant au plus une unité de temps (quantum) avant de passer au processus suivant dans la file d'attente. Ce processus se répétera jusqu'à ce que tous les processus soient complètement exécutés.

P1	P2	P3	P4	P5

P5	P4	P3	P2	P1 =>	CPU
P1	P5	P4	P3	P2 =>fin	CPU

P1	P5	P4	P3=>	CPU
P3	P1	P5	P4=>fin	CPU

P3	P1	P5=>	CPU
P5	P3	P1=>	CPU
P1	P5	P3=>fin	CPU

…

P1	P5=>fin	CPU
P1=>FIN	CPU	

2.Quelle est, de ces trois politiques, celles qui correspond à un temps minimal d'attente moyen par processus ?

La politique qui correspond à un temps minimal d'attente moyen par processus est l'algorithme SJF (Shortest Job First).

Partie 2:

Processus	P1	P2	P3	P4	P5
Duree	6	3	1	5	2
Date d'arrivée	0	1	2	4	8

1)On suppose que l'ordonnancement est fait suivant l'algorithme du plus court temps restant non préemptif. Indiquez dans chaque case du diagramme ci-dessous quel processus est affecté à l'Unité de Contrôle (UC) à chaque pas de temps de l'exécution.

Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
UC	P2	P3	P2	P2	P4	P4	P4	P4	P4	P4	P5	P5	P1	P1	P1	P1	P1

2. On suppose que l'ordonnancement est fait suivant l'algorithme du plus court temps restant préemptif. Indiquez dans chaque case du diagramme ci-dessous quel processus est affecté à l'Unité de Contrôle (UC) à chaque pas de temps de l'exécution.

Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
UC	P2	P3	P2	P2	P4	P4	P4	P5	P5	P4	P4	P1	P1	P1	P1	P1	P1

3.On suppose que l'ordonnancement est fait suivant l'algorithme round robin avec un quantum de temps fixé à 2. Indiquez dans chaque case du diagramme ci-dessous quel processus est affecté à l'Unité de Contrôle (UC) à chaque pas de temps de l'exécution.

Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
UC	P1	P1	P2	P2	P3	P4	P4	P5	P5	P1	P1	P2	P4	P4	P1	P1	P4

4. A partir de maintenant, on définit un algorithme d'ordonnancement à deux niveaux de priorité :

- Le niveau 0 obéit à un ordonnancement round robin quantum 2 : entre eux. Les processus de priorité 0 suivent cet ordonnancement.

  Le niveau 1 obéit à un ordonnancement « plus court d'abord non préemptif (en cas d'égalité, le plus ancien dans la file obtient le processeur)

- Le niveau 0 a priorité sur le niveau 1 : tous les processus de priorité 0 sont toujours prioritaires sur ceux de priorité 1 et ce de manière préemptive.

Les priorités de nos processus sont définies de la manière suivante :

Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
UC	P2	P2	P2	P3	P4	P4	P4	P5	P5	P4	P4	P1	P1	P1	P1	P1	P1

Exercice 3 Odonnancement avec entree sortie E/S
On considère 3 processus, A, B, C dont on suppose que l'exécution se compose d'une répétition de giclées CPU et d'opération d'Ets de longueur constante.

1. Pour A : 7 unités de temps (ut) d’accès CPU puis 2 ut d’E/S, 7 ut CPU, 2 ut E/S, 7 ut CPU, 2 ut E/S, etc.
2. Pour B : 2 ut CPU, 2 ut E/S, 2ut CPU, 2ut E/S, etc.
3. Pour C : 5ut CPU, 4 ut E/S, 5 ut CPU, 4 ut E/S, etc.
4. On supposera que A se présente en premier, suivi de B à I ut puis tard, puis C, à I ut après B

Montrez comment les 3 processus vont utiliser le CPU pendant les 3() unités de temps à venir dans les cas suivants :

1. Les processus n'attendent pas pour leurs E/S car ils ont chacun leur périphérique propre
    1. Selon l'ordonnancent FIFO

PROCESSUS A
Pret * * * * * * * * * *
Elu *
cpu * * * *

Processus B

```
Prêt * * * * * * *         * * * * * * * * * *         * * * * *
Elu                * *                         * *
E/S                    * *                         * *
```

Processus C

```
Prêt * * * * * * * * *
Elu                    * * * * *                 * * * * *
E/S                              * * * * * * * *           * * *
```

2. Selon l'ordonnancent SJF

Processus A

```
Prêt                   * * * * *                   * * * * *
Elu  * * * * * * *               * * * * * * *               * *
E/S                * *                         * *
```

Processus B

```
Prêt * * * * * * *         * * * * * * * * * *         * * * * *
Elu                * *                         * *
E/S                    * *                         * *
```

Processus C

```
Prêt * * * * * * * * *
Elu                    * * * * *                 * * * * *
E/S                              * * * * * * * *           * * *
```

2. Les trois processus utilisent le même périphérique d’E/S dont la file d’attente est géré par SJF et l’accès au CPU est géré selon l’ordonnancement du tourniquet, avec un quantum de 3.

2. Processus A

```
Prêt           *  *  *  *  *
Elu    *  *  *
E/S
```

4. Processus B

```
Prêt * * *           *
Elu        * * *
E/S              * *
```

6. Processus C

Prêt * * * * * * * *
Elu * * *
E/S

Exercice 4 : Résolution d'un problème d'ordonnancement

Un système d'exploitation installé sur une machine disposant d'un seul processus doit gérer 5 processus A, B, C, D et E. Comme illustré dans la figure ci-dessus, chaque processus est décrit par le nombre d'instructions qu'il contient, son temps de lancement, son PID et l'UID de l'utilisateur qui l'a lancé :

Processus	**PID**	**Instructions**	**Temps de lancement**	**UID**
A	**1**	**3**	**1**	**1**
B	**2**	**6**	**2**	**2**
C	**3**	**4**	**3**	**2**
D	**4**	**2**	**4**	**1**
E	**5**	**3**	**6**	**3**

Le processeur de la machine peut traiter l'instruction par cycle, les temps de lancement étant, pour des raisons de simplicité, exprimés eux aussi en cycle de processeurs (un temps de lancement de 2 indique que le programme a été lancé au début du deuxième cycle de processeur).

Une table d'ordonnancement peut être écrite pour un ensemble de processus sous la forme du tableau.

Cycle	Processus en exécution	Etat de fin de cycle	Nombre d'instructions restantes

Ou la colonne cycle contient le numéro du cycle processeur (le premier étant 1), la colonne Processus en exécution contient le processus présent dans le processeur durant le cycle et Etat de fin ce cycle contient l'état du processus à la fin du cycle. Cet état peut être : en attente, en exécution ou arrêté.

1. Ordonnancement Round Robin (Tourniquet)

   En utilisant un ordonnanceur de type Round Robin :

   1. Écrire la table d'ordonnancement du système pour traiter l'ensemble des processus.

Cycle	Processus en exécution	Etat de fin de cycle	Nombre instructions restantes
1	A	2	En attente
2	B	5	En attente
3	C	3	En attente
4	D	1	En attente
5	A	1	En attente
6	B	4	En attente
7	C	2	En attente
8	D	0	Arrêté
9	E	2	En attente
10	A	0	Arrêté
11	B	3	En attente

12	C	1	En attente
13	E	1	En attente
14	B	2	En attente
15	C	0	Arrêté
16	E	0	Arrêté
17	B	1	En attente
18	B	0	Arrêté

2. Calculer pour chaque processus son temps d'attente (nombre de cycle avant la première exécution du processus) et son temps d'exécution total (nombre de cycles entre l'exécution de sa dernière instruction et son lancement).

Processus	Temps d'attente	Temps d'exécution
A	1	9
B	2	17
C	3	13
D	4	5
E	9	8

2. Ordonnancement avec priorité

Nous utilisons maintenant un ordonnanceur basé sur des priorités qui fonctionne de la manière suivante :

· Chaque processus possède une priorité sous forme de nombre entier entre 0 et 10

· La priorité maximale d'un processus et 0, la minimale est 10

· Au lancement, un processus à une priorité de 0

· A chaque cycle d'exécution, la priorité du processus augmente de 1

· A chaque cycle, l'ordonnanceur exécute le processus le plus prioritaire

· A chaque cycle, la priorité des processus non exécuté diminue de 1 (sans pouvoir passer sous 0)

· Chaque fois que 2 processus on même priorité, celui qui au moins d'instructions est choisi en utilisant cet ordonnanceur :

1. Écrire la table d'ordonnancement du système pour traiter l'ensemble des processus (ajouter à la table une colonne priorité qui contient la priorité du processus à la fin du cycle).

Cycle	A	B	C	D	E

2. Calculer pour chaque processus son temps d’attente et son temps d’exécution.

Processus	Temps d’attente	Temps d’exécution
A	0	7
B	0	17
C	0	10
D	0	3
E	3	5

3. Comparer les tables d’ordonnancement des questions 2.1 et 2.2. Quelle est la plus intéressante ?

L'ordonnancement avec priorité favorise l'exécution rapide des petits processus en leur accordant une priorité élevée. En revanche, le round robin, adapté aux gros processus, garantit une équité d'accès au temps CPU, évitant ainsi que les processus longs monopolisent les ressources pendant de longues périodes. Ainsi, chaque approche répond efficacement aux besoins spécifiques des processus en fonction de leur taille et complexité.

4. Combien de cycles sont nécessaires pour traiter tous les processus dans les 2 cas ?

Peu importe l'ordonnancement utilisé, les 18 cycles sont nécessaires pour exécuter intégralement tous les processus. Quel que soit le mode d'attribution de ressources CPU, chaque instruction de tous les processus doit être exécutée, assurant ainsi l'achèvement global en 18 cycles. L'efficacité du planning influe sur la distribution et le temps d'exécution, mais l'ensemble des instructions est toujours exécuté.