Exercice 1 : ordonnancement des processus

A/ Supposons qu'un système d'exploitation gère cinq processus (A, B, C, D, et E) qui arrivent à différents moments et doivent être exécutés sur un seul processeur. Chaque processus a une durée d'exécution différente, et le système utilise l'ordonnancement préemptif en utilisant la priorité comme critère.

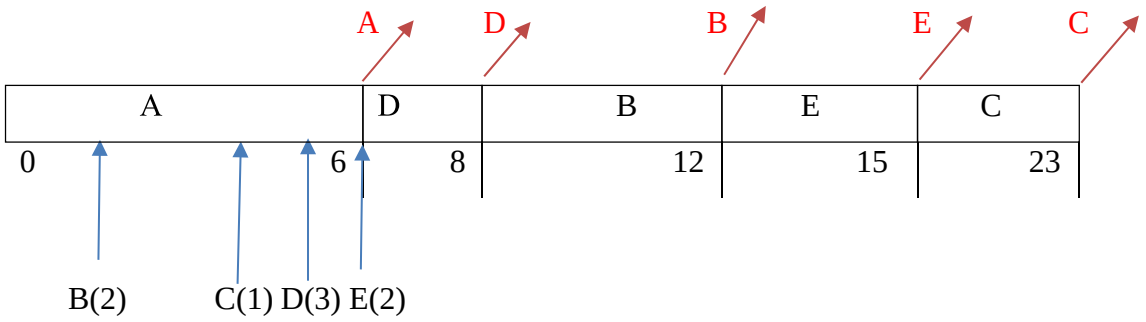
Les caractéristiques des processus sont les suivantes :

Processus	Temps d'arrivée	Temps d'exécution	Priorité
A	0	6	3
B	2	4	2
C	4	8	1
D	5	2	3
E	6	3	2

Remarque : priorité i est supérieure par rapport à la priorité i-1

À chaque instant, le système choisit le processus avec la priorité la plus élevée. En cas d'égalité de priorité, le processus qui arrive en premier est choisi. Si un processus termine son exécution ou est préempté, le processus avec la priorité la plus élevée est choisi.

1. Quel sera l'ordre d'exécution des processus ? Fournissez une séquence d'exécution avec l'heure d'arrivée, la priorité, le temps d'exécution restant, et le processus en cours.



2. Quelle est la durée totale nécessaire pour exécuter les trois processus dans cet ordre ?

   La durée totale nécessaire pour exécuter les trois processus= 23 ut

3. Donnez le temps moyen de séjour (temps moyen de rotation).

   Temps de séjour = temps fin d'exécution – temps d'arrivée

$T_A = 6-0=6$; $T_B = 12-2=10$; $T_C = 23-4=19$; $T_D = 8-5=3$; $T_E = 15-6=9$;

Temps moyen de séjour= (6+10+19+3+9) /5=9.4

B/ Supposons que vous ayez maintenant un système d'exploitation utilisant l'algorithme d'ordonnancement Round Robin. L'intervalle de temps de chaque quantum (temps de rafraîchissement) est de 3 unités de temps.

1. Appliquez l'algorithme Round Robin pour ordonner l'exécution de ces processus. Fournissez une séquence d'exécution montrant l'heure d'arrivée, le temps d'exécution restant pour chaque processus à chaque quantum, et le processus en cours.
2. D E A B C

A (3)	B (1)	C (5)	D (0)	E (0)	A (0)	B (0)	C (0)
0 3	6	9	11	14	17	18	23

3. Quel est le temps d'attente moyen des processus dans cet ordre d'exécution ?

TA_A =(0-0)+(14-3)=11 ; TA_B =(3-2)+(17-6)=12 ; TA_C =(6-4)=2 ; TA_D =(9-5)=4 ; TA_E =(11-6)=5

Temps d'attente moyen =(11+12+2+4+5) /5=6.8

Exercice 2 :

Écrire un programme **P** qui crée deux fils F**1** et **F2**, tous les processus doivent afficher le **PID** des trois processus avant de se terminer proprement.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int pipe_fd[2];
    pid_t pid_F1, pid_F2;
    pid_t p_pid = getpid(); // PID du processus parent

    // Création du tube pour la communication
    if (pipe(pipe_fd) == -1) {
        perror("Erreur lors de la création du tube");
        return 1;
    }

    // Création du premier fils (F1)
    pid_F1 = fork();

    if (pid_F1 < 0) {
        fprintf(stderr, "Erreur lors de la création du processus fils F1\n");
        return 1;
    } else if (pid_F1 == 0) {
        // Code du premier fils (F1)
        close(pipe_fd[1]); // Fermer l'extrémité d'écriture du tube

        // Lire le PID de F2 depuis le tube
        pid_t f2_pid;
        read(pipe_fd[0], &f2_pid, sizeof(pid_t));
        close(pipe_fd[0]); // Fermer l'extrémité de lecture du tube

        // Afficher les PIDs
        printf("Je suis le fils F1 avec PID : %d, PID de mon père : %d, PID du
fils F2 : %d\n", getpid(), p_pid, f2_pid);
        exit(0);
    } else {
        // Création du deuxième fils (F2)
        pid_F2 = fork();

        if (pid_F2 < 0) {
            fprintf(stderr, "Erreur lors de la création du processus fils
F2\n");
            return 1;
        } else if (pid_F2 == 0) {
            // Code du deuxième fils (F2)
            close(pipe_fd[0]); // Fermer l'extrémité d'écriture du tube
            pid_t pidfils2 = getpid();
            // Lire le PID de F1 depuis le tube
            pid_t f1_pid;
            write(pipe_fd[1], &pidfils2, sizeof(pid_t));;
            close(pipe_fd[1]); // Fermer l'extrémité de lecture du tube
```

```
            // Afficher les PIDs
            printf("Je suis le fils F2 avec PID : %d, PID de mon père : %d,
PID du fils F1 : %d\n", getpid(), p_pid, pid_F1);
            exit(0);
        } else {
            // Code du processus parent
            // Attendre la fin des fils
            wait(NULL);
            wait(NULL);

            // Afficher les PIDs
            printf("Je suis le processus parent avec PID : %d\n", getpid());
        }
    }

    return 0;
}
```

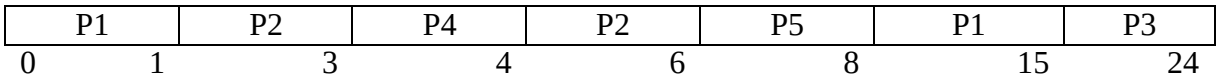
Exercice 3 :

Considérez un système d'ordonnancement de processus avec cinq processus à traiter. Chaque processus est caractérisé par un identifiant unique, un temps d'arrivée, un temps d'exécution et une priorité. Les détails des processus sont les suivants :

Processus	Temps d'arrivée	Temps d'exécution
P1	0	8
P2	1	4
P3	2	9
P4	3	1
P5	4	2

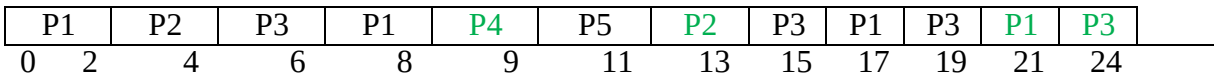
1. Utilisez ces informations pour dessiner le diagramme de Gantt représentant l'ordonnancement des processus à l'aide de l'algorithme SRTF (Shortest Remaining

Time First). Précisez le temps d'attente moyen.



Average Waiting Time:
P1: (0-0)+ (8-1)=7 units
P2: (1-1)+ (4-3)=1 units
P3: (15-2)=13 units
P4: (3-3)=0 units
P5: (6-4)=2 units
Global Average Waiting Time: (7+1+13+0+2)/5= 4.6 units

2. Utilisez les mêmes processus pour simuler l'ordonnancement en utilisant l'algorithme **Round Robin** (**RR**) avec un quantum de temps de **2 unités**. Dessinez le diagramme de Gantt représentant l'ordonnancement des processus.



Average Waiting Time:
P1:(0-0)+ (6-2) +(15-8)+ (19-17)=0+7+4+2= 11 units
P2:(2-1)+ (11-4)=1+7= 8 units
P3:(4-2)+ (13-6)+(17-15)+ (21-19)=2+7+2+2= 13 units
P4:(8-3)= 5 units
P5:(9-4)= 5 units
Global Average Waiting Time: (11+8+13+5+5)/5=8.4 units

3. Comparez les résultats obtenus avec les algorithmes SRTF et Round Robin (RR) pour les processus donnés. Discutez des avantages et des inconvénients de chaque algorithme en termes de temps d'attente moyen, de temps d'exécution total et de réactivité vis-à-visdes processus avec différents temps d'exécution.

SRTF offre un temps d'attente moyen **plus faible** car il priorise les processus avec le temps d'exécution restant le plus court. Cela réduit le temps d'attente des processus courts, en particulier pour P4 qui se termine immédiatement.

- SRTF avec priorité: Priorise fortement les processus courts, minimisant leur temps d'attente.
- Round Robin: Peut entraîner une attente plus longue pour les processus courts en raison du temps d'attente dans la file d'attente des prêts, surtout si le quantum est grand.