EXAMEN de CONTROLE CONTINU

Exercice 1 : QUESTIONS DE COURS (5 pts - 20 minutes)

Répondez brièvement :

1.1 Qu'est-ce qu'une *interruption* ? A quoi sert et comment ça marche une *interruption* ?

1.2 Qu'est-ce qu'une *interruption horloge* ? A quoi sert-elle ?

1.3 Qu'est-ce qu'un *temps partagé* ? A quoi sert-il et comment peut-il être implémenté ?

1.4 Qu'est-ce qu'une *ressource critique* ? Qu'est-ce qu'une *section critique* ? Quels problèmes doit-on éviter ?

1.5 Qu'est-ce qu'un *sémaphore* ? A quoi sert-il ?

Exercice 2 : SYNCHRONISATION (4 pts - 20 minutes)

P1	**P2**	**P3**
afficher "L"	afficher "O"	afficher "V"
afficher "E"		
fin	fin	fin

2.1 On lance P1, P2 et P3 (presque) en même temps dans le même terminal. Quels sont les affichages que l'on peut observer ?

2.2 Modifier les programmes pour que l'affichage soit toujours LOVE.

Exercice 3 : ORDONNANCEMENT (4 pts - 15 minutes)

Soit TS le temps de service d'un travail, c'est-à-dire le temps écoulé entre la soumission du travail et sa fin. On considère un système de traitement séquentiel (batch) dans lequel quatre travaux arrivent dans l'ordre suivant :

NO du travail	Instant d'arrivée	Durée
1	0	8
2	1	4
3	2	9
4	3	5

3.1 Donner le **TS** moyen dans le cas où l'on adopte la politique PAPS (Premier Arrivé, Premier Servi, ou encore FCFS, *First Come First Served*)

3.2 Donner le **TS** moyen dans le cas où l'on adopte la politique préemptive : PCA (le plus court d'abord, ou encore SJN, *Shortest Job Next*)

Exercice 4 : CRÉATION DES PROCESSUS (7 pts - 25 minutes)

Soit le programme suivant:

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

int main(void) {
        pid_t pid;
        int i;

        if ((pid = fork()) == -1) {
                perror("fork");
                exit(1);
        }
        if (pid == 0) { /* fils1 */
                for (i = 1; i <= 50; i++)
                printf("%d\n", i);
                return 0;
        }
        if ((pid = fork()) == -1) {
                perror("fork");
                exit(1);
        }
        if (pid  == 0) { /* fils2 */
                for (i = 51; i <= 100; i++)
                printf("%d\n", i);
                return 0;
        }
        return 0;
}
```

4.1 Donnez l'arborescence des processus créés. Quel est le nombre total de processus créés ?
4.2 Que fait ce programme ? Qu'affiche-t-il ?
4.3 L'ordre d'exécution des différents processus est-il prédictible ? Expliquer.
4.4 Synchronisez les processus pour que l'affichage se fasse toujours en ordre croissant (sans utiliser la commande *sleep*).