# Série 1 – Gestion du processeur central

## Exercice 1

On considère un système monoprocesseur et 4 processus P1, P2, P3 et P4 qui effectuent du calcul et des entrées/sorties avec un disque selon les temps donnés ci-dessous:

| **Processus P1** | **Processus P2** | **Processus P3** | **Processus P4** |
|---|---|---|---|
| Calcul : 3 U | Calcul : 4 U | Calcul : 2 U | Calcul : 7 U |
| E/S : 7 U | E/S : 2 U | E/S : 3 U | |
| Calcul : 2 U | Calcul : 3 U | Calcul : 2 U | |
| E/S : 1 U | E/S : 1 U | | |
| Calcul : 1 U | Calcul : 1 U | | |

On considère que l'ordonnancement sur le processeur se fait selon une politique à priorité préemptive: le processus élu à un instant t est celui qui est à l'état prêt avec la plus forte priorité.
On donne priorité (P1) > priorité (P3) > priorité (P2) > priorité (P4).
On considère que l'ordre de services des requêtes d'E/S pour le disque se fait toujours selon une politique FIFO.
Sur le graphique de la page 2, donnez le chronogramme d'exécution des 4 processus P1, P2, P3 et P4. Justifiez votre raisonnement, en expliquant la gestion des files d'attentes et les transitions des processus. Donnez le temps de réponse moyen obtenu.

## Exercice 2

On considère un système monoprocesseur de type Linux dans lequel les processus partagent un disque comme seule ressource autre que le processeur. Cette ressource n'est accessible qu'en accès exclusif et non préemptif.
Un processus peut être en exécution, en attente d'E/S, en E/S ou en attente du processeur.
Les demandes d'E/S sont gérées à l'ancienneté.
Dans ce système, on considère 4 processus P1, P2, P3, P4 pour lesquels:

- P1 et P2 sont des processus appartenant à la classe **SCHED_FIFO**. Dans cette classe, le processeur est donné au processus de plus haute priorité. Ce processus peut être préempté par un processus de la même classe ayant une priorité supérieure;
- P3 et P4 sont des processus appartenant à la classe **SCHED_RR**. Dans cette classe, le processeur est donné au processus de plus haute priorité pour un quantum de temps égal à 10 ms. La politique appliquée est celle du tourniquet.

Les processus de la classe **SCHED_FIFO** sont toujours plus prioritaires que les processus de la classe **SCHED_RR**.
Les 4 processus ont le comportement suivant:

| **P1:** (priorité 50) | **P2** (priorité 49) | **P3** (priorité 49) | **P4** (priorité 49) |
|---|---|---|---|
| Calcul (40 ms) | Calcul (30 ms) | Calcul (40 ms) | Calcul (100 ms) |
| Lecture disque (50 ms) | Lecture disque (80 ms) | Lecture disque (40 ms) | |
| Calcul (30 ms) | Calcul (70 ms) | Calcul (10 ms) | |
| Lecture disque (40 ms) | Lecture disque (20 ms) | | |
| Calcul (10 ms) | Calcul (10 ms) | | |

Établissez le chronogramme d'exécution des 4 processus.

## Exercice 3

On considère un système dans lequel les seules ressources partagées sont un disque géré par un canal et un processeur. Les requêtes disques sont gérées à l'ancienneté (FIFO).
L'ordonnanceur est activé à chaque début et fin de traitement d'une E/S disque ou après un quantum Q si une E/S ne s'est pas produite dans le dernier quantum.
L'ordonnanceur alloue le processeur au processus Pi en attente qui a le plus fort rapport **Ti/Tcpui**.

- **Ti** représente la durée totale écoulée depuis le début de l'exécution du processus Pi (c'est-à-dire le temps écoulé depuis l'instant de première requête du processeur par le processus Pi).
- **Tcpui** est le cumul des durées pendant lesquelles le processeur a été alloué au processus i.

Lorsque le rapport **Ti/Tcpui** est égal à 0/0, celui-ci est interprété comme +∞.
Lorsque plusieurs processus ont la même priorité, c'est le processus d'indice le plus fort qui obtient le processeur.
L'ordonnanceur utilise un délai de garde (temporisation) de durée q. Il est activé:

- soit lorsque le délai est écoulé;
- soit lorsque le processus actif fait une requête d'entrée-sortie;
- soit lorsque l'exécution d'une telle requête se termine.

Dans tous les cas, il commence par réarmer le délai de garde puis procède à l'allocation quand celle-ci est possible.
A l'instant initial trois processus sont présents dans le système et commencent à s'exécuter en faisant leur première requête à l'ordonnanceur.
Le tableau suivant donne (en nombre de quantums) la durée totale de CPU nécessaire à leur exécution, le nombre d'entrée-sortie disque réalisé par chaque processus et la date, mesurée en temps CPU écoulé depuis le début d'exécution, des requêtes d'entrée-sortie.

| N° processus | Temps total CPU (quantum) | Nombre d'accès disque | Dates des accès (tps CPU relatif) |
|---|---|---|---|
| 1 | 5,5 | 1 | 1,5 |
| 2 | 5 | 0 | -- |
| 3 | 1,5 | 2 | 0,5 puis 1 |

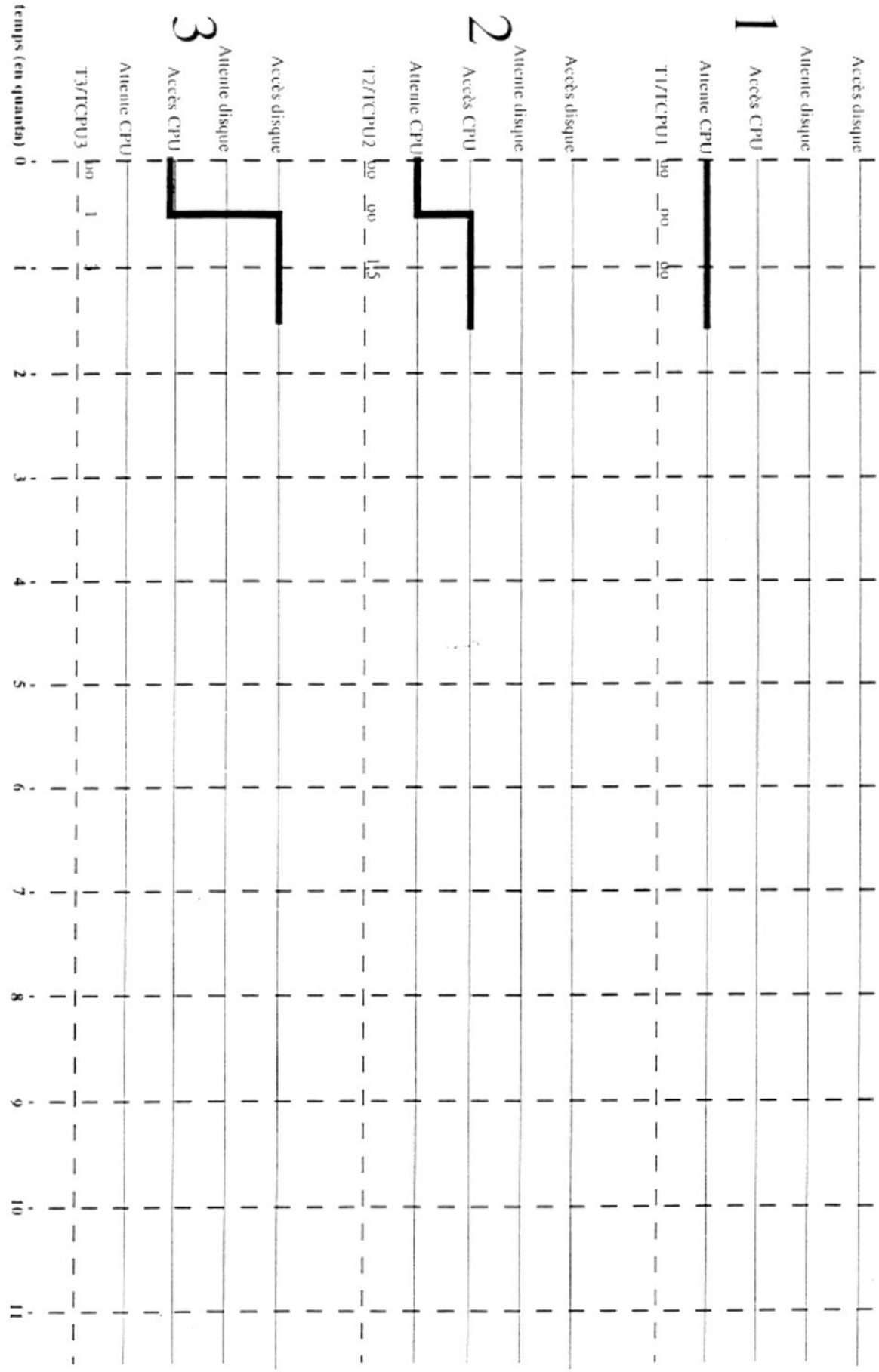
La durée d'un service disque est deux quantums et les demandes sont servies à l'ancienneté (FIFO).

1. Compléter le chronogramme de la page 4 qui donne l'état des différents processus à chaque instant ainsi que la valeur des rapports **Ti/Tcpui** lors de chaque changement d'état.
2. Commenter le chronogramme en donnant les avantages et inconvénients de cette gestion du processeur.

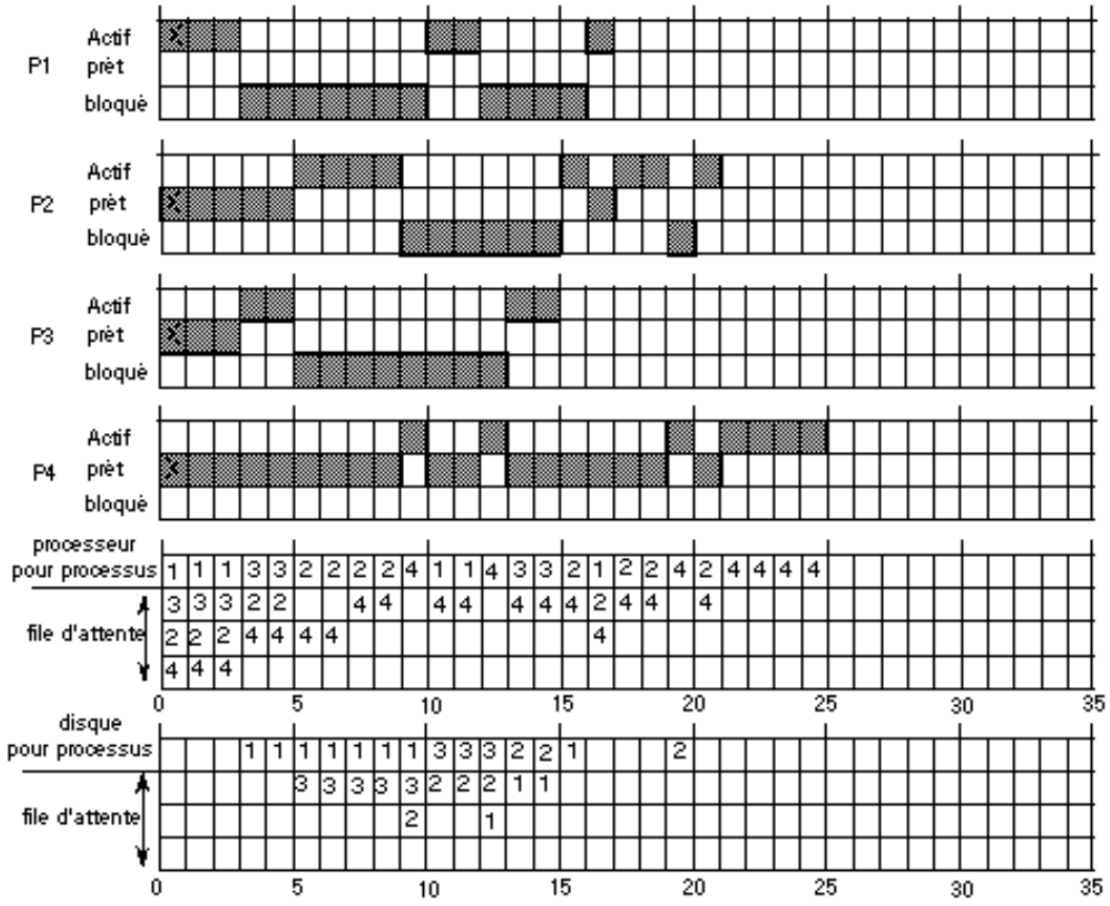
## Exercice 4:

Les processus demandeurs du processeur central sont rangés dans N files $F_1$, $F_2$,...,$F_N$.
A chaque file est associé un quantum de temps $Q_i$ et les processus arrivent par la file $F_1$.
Un processus situé en tête de file $F_i$ (i>1) ne sera pris en compte que si toutes les files $F_j$ (0<j<i) sont vides.
Si un processus pris de la file $F_i$ ne s'est pas terminé à la fin de son quantum, il sera placé dans la file $F_{i+1}$, sauf pour $F_N$ où il y restera jusqu'à sa terminaison.
Réaliser cette politique dans le cas où :

- N=4
- $Q_{i+1}=Q_i*2$ et que $Q_1$=20ms
- Interruption horloge à chaque 20ms


Accès disque
Attente disque
1
Accès CPU
Attente CPU
T1/TCPU1
Accès disque
Attente disque
2
Accès CPU
Attente CPU
T2/TCPU2
Accès disque
Attente disque
3
Accès CPU
Attente CPU
T3/TCPU3
temps (en quanta) 0 1 2 3 4 5 6 7 8 9 10 11

**Solution Exercice 1 :**

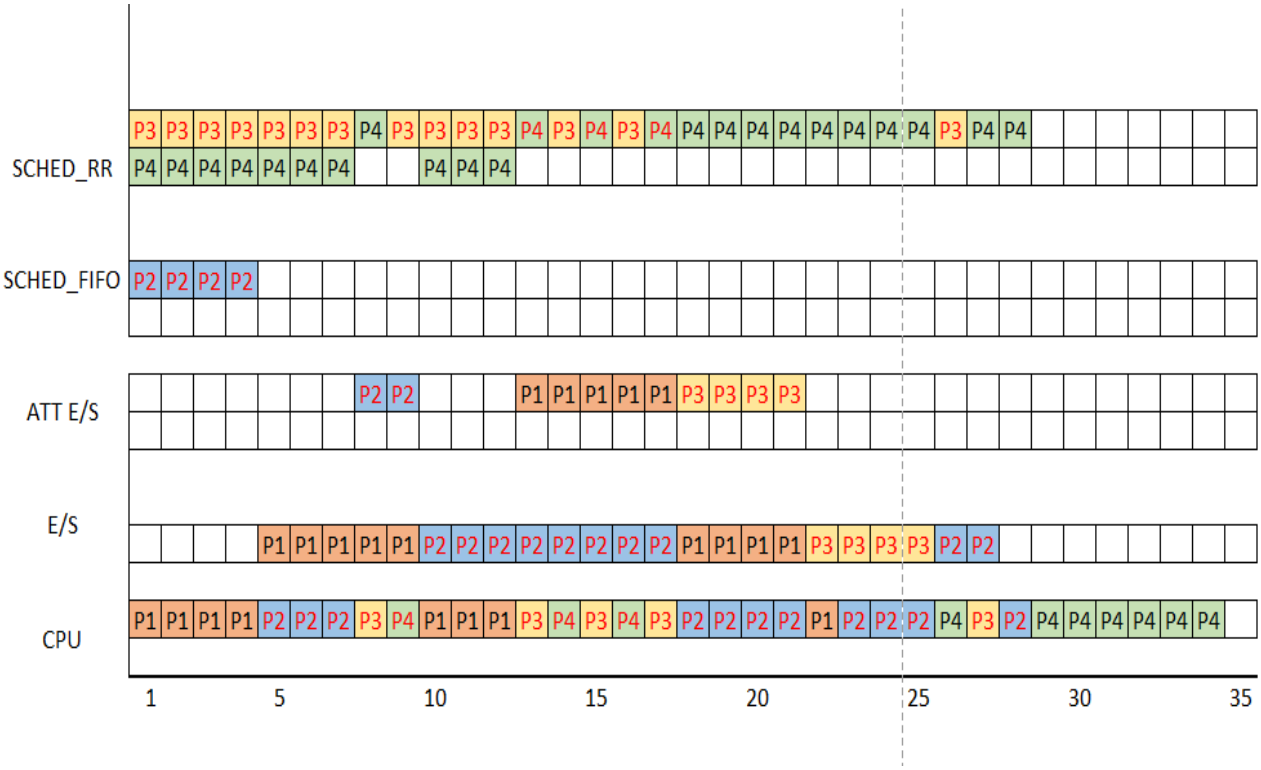



***Notice explicative du déroulement:***

À 0, P1 est actif et obtient le processeur pour 3 UT (fin en 3).
À 3, P1 accède au disque, qui était libre évidemment, pour 7 UT (fin en 10). P3 devient actif et obtient le processeur pour 2 UT (fin en 5).
À 5, P3 passe en tête de file du disque, et P2 devient actif pour 4 UT (fin en 9).
À 9, P2 passe en deuxième de la file disque et P4 devient actif pour au plus 7 UT (fin ≤ 16).
À 10, l'entrée-sortie de P1 se termine et P1 devient prêt, mais étant de priorité supérieure à celle de P4, devient actif pour 2 UT (fin en 12), et P4 passe en tête de la file du processeur (en 1). L'entrée sortie de P3 commence pour 3 Ut (fin en 13).
À 12, P1 passe en queue de file disque (en 2) et P4 devient actif pour au plus 6 UT (fin ≤ 18).
À 13, l'entrée-sortie de P3 se termine et P3 devient prêt, mais étant de priorité supérieure à celle de P4, devient actif pour 2 UT (fin en 15), et P4 passe en tête de la file processeur. L'entrée-sortie de P2 commence pour 2 UT (fin en 15).
À 15, P3 se termine et l'entrée-sortie de P2 se termine. P2 devient prêt, mais étant de priorité supérieure à celle de P4, devient actif pour au plus 3 UT (fin ≤ 18), et P4 passe en tête de la file processeur. L'entrée-sortie de P1 commence pour 1 UT (fin en 16).
À 16, l'entrée-sortie de P1 se termine et P1 devient prêt, mais étant de priorité supérieure à celle de P2, devient actif pour 1 UT (fin en 17). P2 est en 1 et P4 en 2 de la file processeur.
À 17, P1 se termine, P2 devient actif et obtient le processeur pour 2UT (fin en 19).
À 19, P2 accède au disque pour 1 Ut (fin en 20) et P4 devient actif pour au plus 5 UT (fin ≤ 24).
À 20, l'entrée-sortie de P2 se termine et P2 devient prêt, mais étant de priorité supérieure à celle de P4, devient actif pour 1 UT (fin en 21), et P4 passe en tête de la file processeur.
À 21, P2 se termine et P4 devient actif pour 4 UT (fin en 25).
À 25, P4 se termine et il n'y a plus de processus.

***Le temps de réponse*** de P1 est de 17, celui de P2 est de 21, celui de P3 est de 15 et celui de P4 est de 25. Le total est 78, soit une moyenne de 19,5 UT.

**Solution Exercice 2 :**




**Solution Exercice 4 :**

1/ **SVC( cause,…)**

<Sauvegarder Context>

Switch (cause):

Case Create: Si (resource Disponible )

Alors allouer les ressources;

Créer PCB (Pi)

PCB.id= id;

PCB.état=Prêt;

PCB.CLassPrio(F0);

Enfiler($F_0$, PCB(Pi));

Si (CPU libre) alors call Scheduler() ;fsi ;

Sinon Traiter_erreur() ;

Fsi ;

Terminer : Libérer les ressources ;

Suprimer PCB(Pi) ;

Call Scheduler()

Case Dem E/S: Defiler (P, Actif);

P.état = Bloqué;

Enfiler(P, Bloquer)

Si (∃ perifph && droit d'accès)

Alors call init_ent();

Call Scheduler();

Sinon Traiter Erreur;

Fsi;

Fin Case

Case Fin E/S:

<Restorer Le context>

**Scheduler(){**

Répéter jusqu' à FAUX:

Si (7vide F1) alors PCB=Défiler (F1)

Sinon si (7vide F2) alors PCB=Défiler (F2) ;

Sinon si (7vide F3) alors PCB= Défiler (F3) ;

Sinon si (7vide F4) alors PCB =Défiler (F4) ;

Fsi ;

Si (PCB !=Nill)

PCB.état=actif

Enfiler (Actif, PCB)

Timer=0 ;

LPSW(tete(actif)) ;

Sinon action wachdog() ;fsi ;}

**RIT Fin E/S**

{<Sauvegarder Context>

Si (erreur) alors traiter erreur;

Sinon si fin de transfert de tous les caractères

Alors PCB= Défiler(Bloqué)

PCB.état= Prêt

Si (q==0 ) alors Enfiler(PCB.Classe+1,PCB)

Sinon Enfiler (PCB.Classe, PCB)

Si file Bloqué 7vide alors lancer prochaine demande E/S;

Sinon état (périph)=libre ;

Fsi

<Restorer Le context>}

Hmax=20ms

Q1=20 ; Q2=40 ; Q3=80 ;Q4=160

**RIT Horloge()**

{ <Sauvegarder Context>

Timer+=Hmax ;

si (Timer==Q1 && PCB. Classe==F1) alors PCB= défiler (file.actif)

PCB.état=prêt ;

Enfiler(F2, PCB) ;

Call scheduler() ;

Sinon si (Timer==Q2 && PCB. Classe==F2) alors PCB= défiler (file.actif)

PCB.état=prêt ;

Enfiler(F3, PCB) ;

Call scheduler() ;

Sinon si (Timer==Q3 && PCB. Classe==F3) alors PCB= défiler (file.actif)

PCB.état=prêt ;

Enfiler(F4, PCB) ;

Call scheduler() ;

```
        Sinon si (Timer==Q4 && PCB. Classe==F4) alors PCB= défiler (file.actif)
                                                    PCB.état=prêt ;
                                                    Enfiler(F4, PCB) ;
                                                    Call scheduler() ;
        Fsi;
<Restorer Le context>}
```