4.5 : **code complet : en Pseudo C**

```
entrer (pid,f); // ajouter le processus pid a f .

premier (f) -> pid ; // return le tete de la file .

sortir (f) -> pid  ;// return le queue de la file .

vide (f) -> booléen  ;// return true si la file est vide .

struct descripteur_de_processus
{
    reg ; //registres généraux
    psw ; // mot d'état de
    programme
    etat ; // élu, éligible ou bloqué
    prio ; // priorité
    liens ;
}

struct descripteur_de_semaphore
{
    cpt ; // valeur du semaphore
    file ; // file du semaphore
}

// Changement d'état

void ranger_proelu()
{
    proelu->reg =reg ;
    proelu->psw = ancien_mep ;

    /* le mot d'état du
    programme interrompu n'est plus dans mep, mais
    a été rangé dans ancien_mep par le matériel

    */
}


void lancer_exec(proelu)
{
    reg = proelu->reg ;
    ancien_mep = proelu->psw ;
```

```
    rti ; //relance l'exécution à partir de ancien_mep
}


// Traitant de l'interruption horloge

void traitant_it_horloge ()

{
    ranger_proelu() ; /* sauvegarde registres et mep */

    proelu->etat = eligible ;

    entrer (proelu,f_eligibles) ;

    lancer_processus_suivant() ;
}



// Définition et lancement du  prochain processus

void lancer_processus_suivant()
{
    proelu = sortir (f_eligibles) ;
    proelu->etat = elu ;
    lancer_horloge(quantum) ;
    lancer_exec(proelu) ;
}



//  Traitant de l'appel système

void traitant_svc()
{
    // seuls P et V sont possibles
    switch (r0) // r0 code de l'appel système
    {
    case CODE_P : executer_p (r1) ; break ;
    case CODE_V : executer_v (r1) ; break ;
    default : exec_erreur ;
    }
}
```

```
// Traitement de P(s)
void executer_p(s)
{
    s->cpt -- ;

    if (s->cpt >=0 )
    {
        rti ;
    }
    else
    {
        ranger_proelu() ;

        proelu->etat = bloque ;

        entrer (proelu,s->file) ;

        lancer_processus_suivant() ;
    }
}


// Traitement de V(s)

void executer_v(s)
{
    ranger_proelu() ;

    proelu->etat = eligible ;

    entrer (proelu,f_eligibles) ;

    s->cpt ++ ;

    if (s->cpt <=0 )
    {
        paux = sortir (s->file) ;

        paux->etat = eligible ;

        entrer (paux, f_eligibles) ;
    }

    lancer_processus_suivant() ;
```

```
}



int main(void)
{
    // Initialisation :
    vect_int[SVC]=(&traitant_svc,maître, masqué) ;

    timer = 0 ;

    vect_int[HORLOGE]=(&traitant_it_horloge, maître, masqué) ;

    /* il faut initialiser aussi les files de processus */


    return 0 ;
}
```