Corrigé du Devoir2- sys02
Gestion du processeur central

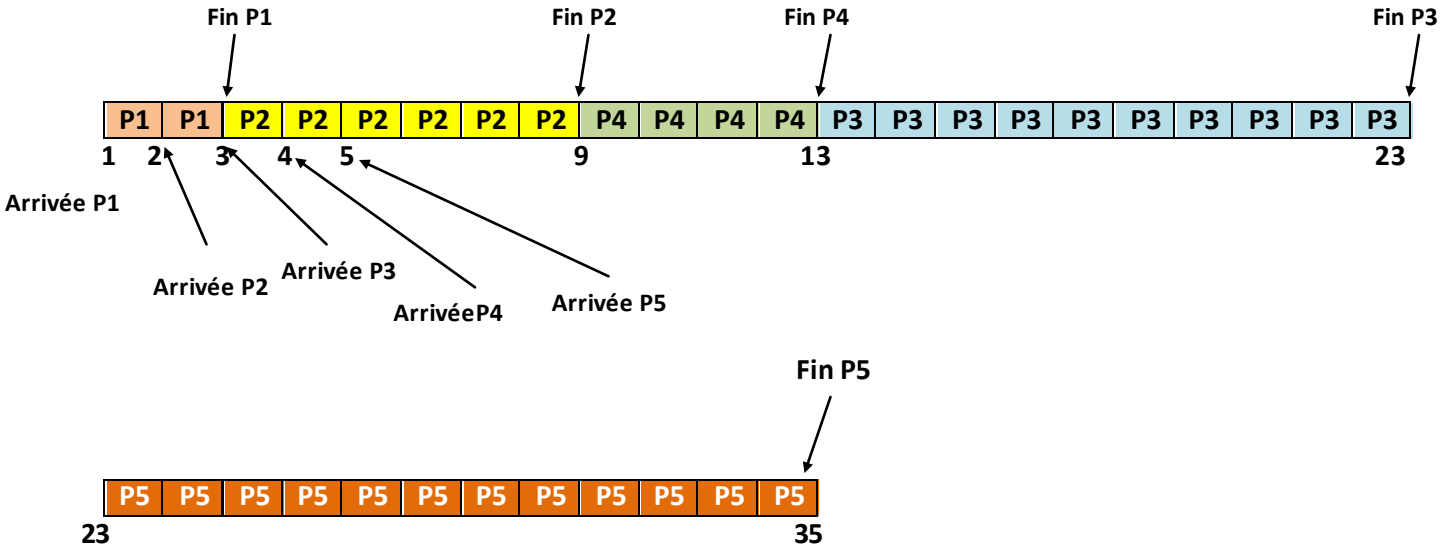
Exercice 1

Remarque Pour cet exercice, on traite le cas de la poltique **SJF** (sans préemption), procéder de la même manière pour les autres politiques

Processus	Ordre de placement dans la file des prêts	Temps d'exécution	Priorité
P1	1	2	2
P2	2	6	4
P3	3	10	3
P4	4	4	5
P5	5	12	1

Politique SJF (Shorted Job First) sans préemption

Dans ce cas, il faut se baser sur la durée d'exécution de chaque processus



Temps de rotation = temps de fin d'exécution– temps de début d'exécution
Temps de répons = temps de fin d'exécution– temps d'arrivée
Temps d'attente = Temps de réponse – durée d'exécution

Trot-moyen = ((3-1)+(9-3)+(23-13)+(13-9)+(35-23)) = (2+6+10+4+12)/5 = **6,8**
Dans ce cas, le temps de rotation de chaque processus est égal à la durée d'exécution car il n y pas de préemption
Tréponse-moyen = ((3-1)+ (9-2)+(23-3)+(13-4)+(35-5)) = (2+7+20+9+30)/5 = **13,6**
Tattente-moyen = ((2-2)+ (7-6)+(20-10)+(9-4)+(30-12)) = (0+1+10+5+18)/5 = **6,8**

Exercice 2

L'algorithme de scheduling des processus du système UNIX est de la classe des schedulers du type ROUND ROBIN avec plusieurs niveaux dépendants. Le scheduler alloue le processeur à un processus pour un quantum de temps. Il préempte le processus qui a consommé son quantum de temps et il le met dans l'un des niveaux de priorités (la file adéquate). La priorité d'un processus est fonction de l'utilisation récente (temps) du processeur central.

a) Quelles sont les informations nécessaires pour implanter cette politique de scheduling?

b) Soient *pr* la priorité d'un processus et *cpu* l'utilisation récente (temps) du processeur central. L'horloge interrompt le processeur central 60 fois à la seconde. A chaque interruption, on incrémente de 1 la valeur de *cpu* du processus actif.

Après un quantum de temps fixé à 1 seconde, le scheduler calcule pour chaque processus la nouvelle valeur *cpu* en utilisant l'équation $cpu = cpu/2$. Après cela, il recalcule *pr* en utilisant l'équation $pr = (cpu/2 + k)$ où k est une constante connue par le système.

- Ecrire les algorithmes nécessaires à cette politique de scheduling dans le cas où le système gère *m* niveaux de priorité et le processus élu est celui qui possède la plus basse priorité.

c) Soient trois processus A, B et C dans le système ayant une priorité initiale de 60. La valeur de la constante k est 60. Après chaque quantum, on recalcule les valeurs de *cpu* et *pr* selon le processus décrit dans la politique de scheduling.

- Appliquer la politique su-décrite en utilisant les valeurs données pour une période de 5 secondes.

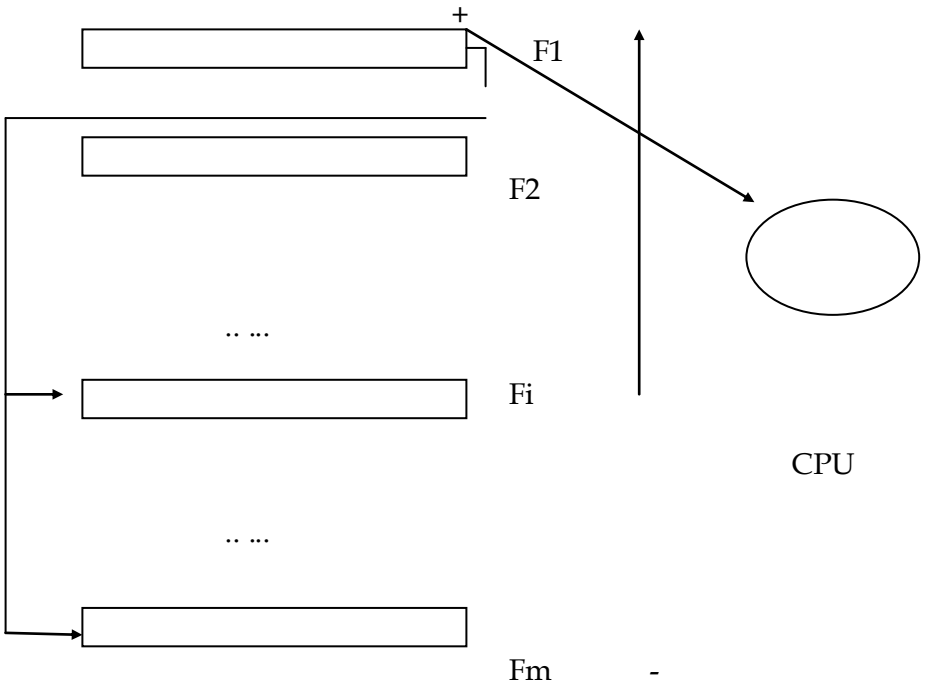
Solution

a. Informations nécessaires :
- Nombre de Files et priorité associée à chaque file.
- Etat des files
- Valeur du Quantum.
- Priorité initiale des processus.
- Formule de calcul de la priorité après un quantum de temps.
- Structure du PCB : Nom, Id, Etat, mot_etat, pri (priorité), Cpu (Temps CPU consommé), svt (suivant).
- Le pointeur Pactif
- Une variable *ind* (= 0 ou 1) pour indiquer s'il faut recalculer les priorités des processus (fin d'un quantum
- La valeur de k

b. Hypothèses

La politique de scheduling utilise ***m*** niveaux (files) de priorité, où Fi correspond à la file avec une priorité égale à i. Les processus d'une même file Fi ont la même priorité i.

La plus basse priorité correspond à la file la plus prioritaire. Donc, F1 est la plus prioritaire.



Pour écrire les algorithmes de scheduling, les valeurs numériques de l'exemple donné en c)- vont être utilisées, càd :

- Quantum = 1 seconde.
- Interruption horloge 60 fois / seconde.
- A chaque interruption horloge, le champ Cpu du processus actif est incrémenté de 1.
- Après un quantum de temps, la formule de calcul de priorité est comme suit :
  Pour chaque processus (dans chaque file), faire :

  - Cpu = Cpu/2.
  - Priorité = Cpu/2 + 60 (nouvelle valeur de Cpu).

Structure du mot d'état PSW

PSW = < masque it horloge, masque it periph, masque it svc, adresse programme>

Procédure Init ()

```
Début
   PSW(Sched) = <1,1,1,adr-sched>;
   PSW(Rit-horloge) = <1,1,1,adr-rit_h>;
   PSW(Rit-FinES) = <1,1,1,adr-it_FinES>;
   PSW(SVC) = <1,1,1, adr-svc>;
   T= période;  // initialiser la période
   Armer_horloge();
   Pactif = nil ;
   k = 60 ;
   ind = 0 ;
   Lpsw(Sched) ; // lancer Scheduler
Fin ;
```

Procédure Calcul-Priorité()

```
Début
// le champ Priorité est égal au N°file
// calcul de la valeur cpu et Pr de tous les processus
    Pour i=1 à m
      Faire
        p=tête(Fi);
        Tant que (p <> nil)
        Faire p.cpu = p.cpu /2;
              p.priorité = (p.cpu/2) + 60;
              p = p->svt;
        Fait;
      Fait;
/*Chaque processus ayant changé de priorité va être remis dans la file correspondant à sa nouvelle priorité */
     Pour i=1 à m
      Faire
       p=tête(Fi);
       Tant que (p <> nil)
       Faire num =  p.priorité;
             Si (num <> i)
               alors  supprimer (Fi, p) ;
                      Enfiler (p, Fnum);
              Fsi;
               p= p->svt;
        Fait;
      Fait;
   Fin ;
```

Scheduler ()

```
Début
Si (ind =1)
  alors
    Calcul-Priorité() ;
// recalculer la  priorité des processus
  dans les files Fi
Fsi ;
// parcourir les files en commençant
   par la plus prioritaire

Etiq :    Pour i = 1 à m
            Faire
             Si (non vide(Fi))
               alors défiler(Fi, Pactif);

               Pactif->etat= "actif";
               Q=60;
               ind = 0 ;
// sera remis à 1 à la fin  d'un
  quantum
               Armer-Horloge() ;
               Lpsw (Pactif);
Fsi ;
Fait ;
Goto Etiq;
Fin ;
```

SVC (cause, nom, adr, ….)

```
<Save GrdCtxt>

Switch (cause)
{ Case : arrivée d'un nouveau processus
  P = CréerPCB(nom, adr);
  Si (Pactif=nil) // aucun processus actif
    alors
          Lpsw(Sched);
    sinon si (p.priorité < Pactif.priorité)
```

Fonction CréerPCB (nom, adr) : PCB

```
Début
P=allouer(PCB);
P.nom=nom;
P.etat=prêt;
P.cpu = 0 ;  // initialiser le temps cpu
P.psw=<0, 0, 0, adr>;
P.Priorite=générer_priorite();
Enfiler(Fi,p);  //Fi est la file correspondant à la
                priorité du processus
```

```
        alors Pactif.etat = prêt;
                    Enfiler(Fi, Pactif);
              //Fi est la file correspondant à la
                      priorité du processus
                Pactif = nil ;
                Lpsw(sched);
/* pour lancer le nouveau process qui est plus
prioritaire*/
            Fsi ;
Fsi;

Case: Fin d'un processus
    Libérer_Ressources (Pactif);
    Pactif = nil ;
    Lpsw(Sched) ;

Case : demande d'E/S
    // non considéré dans cet exercice
    …
    } ;
<Restaure GrdCtxt>
Fin ;
```

```
    Retourner P ;
    Fin


Rit_horloge()
 Début
<Save GrdCtxt>

Pactif.cpu = Pactif.cpu+1 ;
       // incrémenter le temps cpu du processus actif
    Q= Q-1;
    Si (Q=0) // Fin quantum
      alors
          ind =1;  // pour indiquer au scheduler la
                      fin du quantum
          Pactif.etat= "prêt"; // à la fin du quantum
          Enfiler(Fp, Pactif) ;
        Pactif = nil;
        Lpsw(Sched) ;
    Fsi ;

    <Restaure GrdCtxt>
Fin ;
```

c- Exemple numérique

Temps (secondes)	A		B		C		Remarque
	cpu	**pr**	**cpu**	**pr**	**cpu**	**pr**	
0	0 1 2 ... 60	60	0	60	0	60	**Pactif =A** (selon la priorité + l'ordre d'arrivée)
1	30	30/2+60 **=75**	0 1 ... 60	0/2+60 **=60**	0	**60**	**Pactif = B**
2	15	15/2+60 **=67**	30	**75**	0 1 ... 60	**60**	**Pactif = C**
3	7 8 ... 67	**63**	15	**67**	30	**75**	**Pactif =A**
4	33	**76**	7 8 ... 67	**63**	15	**67**	**Pactif = B**
5	16	**68**	33	**76**	7 8 ... 67	**63**	**Pactif = C**

Exercice 3

Soit un système de scheduling basé sur le principe de temps partagé avec priorité. On s'intéresse à la gestion de ce système en utilisant une file d'attente **Fp**, ordonnée par priorité avec un quantum de temps égal à 50ms, et une file **Fa** des processus en attente d'E/S sur un seul organe d'E/S.

- L'arrivée d'un processus dans le système est introduite dans cette file avec une priorité nulle.
- Le processus actif peut restituer le processeur avant la fin de son quantum pour diverses raisons : une demande d'E/S, la fin E/S d'un processus plus prioritaire, la fin totale du programme.
- A chaque période de temps **T**, on recalcule la priorité des processus en attente d'E/S, de telle manière à augmenter la priorité.
- Tout processus qui arrive au bout de son quantum deux fois de suite, il sera inséré dans la file **Fp** avec une priorité nulle.
- Le système dispose d'une horloge de fréquence de 5ms.

1. Quel est le but fixé par ce scheduling ?
2. Indiquer les informations nécessaires pour implanter cet ordonnanceur ?
3. Ecrire les programmes nécessaires pour la mise en œuvre de la politique de scheduling décrite ci-dessus.
4. Est-ce que le système réalise un partage équitable de CPU. Sinon, que faut-il faire pour atteindre cet objectif ?

Solution

1. En supposant que le processus le plus prioritaire est celui de faible valeur de priorité, donc la priorité **0** correspond au processus le plus prioritaire. Donc le but fixé par cette politique de scheduling est de donner *la priorité aux processus orientés calcul*.

2. Informations nécessaires pour implanter cet ordonnanceur ?
   - Les files des processus prêts *Fp* et des processus bloqués (en attente d'E/S) *Fa*,
   - le pointeur du processus actif *Pactif*,
   - la structure de *PCB* définie par les champs suivants : pid, nom, mot-etat, état, priorité, *nbExec* (nombre d'exécutions)
   - la valeur du quantum *Q*, et
   - la valeur de la période *T*.

3. Les programmes nécessaires pour la mise en oeuvre de la politique de scheduling décrite ci-dessus.

Structure du mot d'état PSW

PSW = < masque it horloge, masque it periph, masque it svc, adresse programme>

```
Procédure Init ()
Début
PSW(Sched) =<1,1,1,adr-sched>;
PSW(Rit-horloge) =
            <1,1,1,adr-rit_h>;
PSW (Rit-FinE-S) =
            <0,1,1,adr-rit_FinE-S>;
PSW(SVC) =<1,1,1,adr-svc>;
T=période;  // initialiser la période
Armer_horloge();
Pactif = nil ;
Lpsw(Sched) ; // lancer Scheduler
Fin ;
```

Scheduler ()

```
Début
    //On suppose Fp triée par priorité
Etiq:
      Si (non vide(Fp))
      alors défiler(Fp, Pactif);
           Pactif->etat= "actif";
           Q=50;
Lpsw (Pactif);
Fsi ;
Goto Etiq;
Fin ;
```

SVC (cause, nom, adr, ….)

```
<Save GrdCtxt>

Switch (cause)
{ Case : arrivée d'un nouveau processus
  P = CréerPCB(nom, adr);
  Si (Pactif=nil) // aucun processus actif
    alors
        Lpsw(Sched);
    sinon si (p.priorité < Pactif.priorité)
        alors Pactif.etat = prêt;
              Enfiler-triée (Fp, Pactif);
              Lpsw(sched);
/* pour lancer le nouveau process qui est plus
prioritaire*/
          Fsi ;
Fsi;

Case: Fin d'un processus
    Libérer_Ressources (Pactif);
    Pactif = nil ;
    Lpsw(Sched) ;

Case : demande d'E/S
    Pactif.etat="Bloqué" ;
    Enfiler(Pactif, Fa) ;
    Init_pilote (adr, N) ;
    Pactif = nil ;
    Lpsw(Sched);
    …
    } ;
<Restaure –GrdCtxt>
Fin ;
```

Fonction CréerPCB (nom, adr) : PCB

```
    Début
    P=allouer(PCB);
    P.nom=nom; P.etat=prêt;
    P.psw=<0, 0, 0, adr>;
    P.nbExec=0;
    P.Priorite=générer_priorite();
    Enfiler-triée(Fp,p);
    Retourner P ;
    Fin
```

Rit_horloge()

```
Début
<Save GrdCtxt>
    T=T-5;     // 5 est la fréquence de l'horloge
    Q= Q-5;
    Si (T=0)
      alors Recalculer_priorite(Fa) ;
    Si (Q=0) // Fin quantum
      alors
          Pactif.nbExec++;
          Si (Pactif.nbExec=2)
              alors Pactif.priorité =0 ;
                  Pactif.nbExec=0;
          Fsi;
          Pactif.etat= "prêt"; // à la fin du quantum
          Enfiler-triée(Fp, Pactif) ;
          Pactif = nil;
          Lpsw(Sched) ;
    Fsi ;
    <Restaure –GrdCtxt>
Fin ;
```

Routine Fin d'E/S()

```
Début
<Save Grd-Ctxt>
Si (non Fin d'E/S)
   alors //relancer prochain caractère
sinon
    Défiler(Fa, p) ; // p est le process ayant terminé son E/S
p.etat ="prêt" ;
Enfiler-triée(Fp, p);
Si (non vide (Fa)
alors
    défiler (FDES, DemE-S)
   //Lancer_demandeE/S() ;
Fsi ;
Si (p.priorité < Pactif.priorité)
            //p est plus prioritaire que Pactif
alors pactif = nil ;
       Lpsw(Sched);
/* pour lancer le nouveau process qui est plus prioritaire*/
Fsi ;

<Restaure –GrdCtxt>

Fin ;
```

Procédure Recalculer_priorite (Fa)

```
Début
P = TeteFile (Fa)
Tant que (p <> nil)
        Faire
            p.priorité = p.priorité+1 ;
            p = p-> suivant ;
        Fait ;
Fin ;
```

4/ Le partage est équitable uniquement entre les processus orientés calcul.

L'équité serait satisfaite si la priorité était calculée proportionnellement au temps d'attente dans la file des processus bloqués sur une E/S.