20/12/2023

Systèmes concurrents

ADA 95

Christoph Samuel, Mouche Valentin

Partie 1: RdV ADA

1. Lecteurs/Rédacteurs

1.1. Priorité aux lecteurs

Dans ce premier cas, les **lecteurs ont la priorité** lors de l'accès à une **ressource partagée.** Les lecteurs peuvent lire **simultanément** sans bloquer d'autres lecteurs, mais les **rédacteurs doivent attendre** que tous les lecteurs aient fini avant de pouvoir accéder à la ressource. Cette méthode **évite que les rédacteurs soient bloqués** indéfiniment par des lecteurs en continu.

```
8     -- Interface  Magasinier
9     task type Magasinier is
10       entry debut_lect;
11       entry fin_lect;
12       entry debut_red;
13       entry fin_red;
14    end Magasinier;
```

interface Magasiner

```
33          -- début lecture
34          when (nbred = 0) => accept debut_lect do
```

condition entree debut_lect

```
41          -- début écriture
42          when (nblect + nbred + debut_lect'Count = 0) => accept debut_red do
```

condition entree debut_red

On **définit les entrée** (entry) pour chaque cas debut_lect, debut_red, fin_lect et fin_red dans **l'interface Magasinier** ainsi que les **conditions entree pour debut_lect et debut_red** de sorte à **laisser la priorité** dans notre cas **aux lecteurs.**

```
with TEXT_IO;
use TEXT_IO;

-- un lecteur et un redacteur avec un tampon de taille N
procedure prio_lecteur is
   package int_io is new Integer_io(integer);
   use int_io;

   -- interface Magasinier
   task type Magasinier is
      entry debut_lect;
      entry fin_lect;
      entry debut_red;
      entry fin_red;
   end Magasinier;
```

```
M : Magasinier;

-- interface lecteur
task type lecteur is
end lecteur;

-- interface redacteur
task type redacteur is
end redacteur;

-- body Magasinier
task body Magasinier is

   nblect, nbred, nblectatt : Integer := 0;

begin
   loop
      select

         -- début lecture
         when (nbred = 0) => accept debut_lect do
            nblect := nblect + 1;
            put_line("debut lect");
         end debut_lect;

      or
         -- début écriture
         when (nblect + nbred + debut_lect'Count = 0) => accept debut_red do
            nbred := nbred + 1;
            put_line("debut red");
         end debut_red;

      or
         -- fin lecture
         accept fin_lect do
            nblect := nblect - 1;
            put_line("fin lect");
         end fin_lect;

      or
         -- fin écriture
         accept fin_red do
            nbred := nbred - 1;
            put_line("fin red");
         end fin_red;

      end select;
   end loop;
end Magasinier;

-- corps lecteur
task body lecteur is
   value : Integer := 1;
begin
```

```
        for i in 1..3 loop
            M.debut_lect;
            put_line(" *" & Integer'Image(i) & " *");
            M.fin_lect;
        end loop;
    end lecteur;

    -- corps redacteur
    task body redacteur is
        value : Integer := 0;
    begin
        for i in 1..3 loop
            M.debut_red;
            put_line(" *" & Integer'Image(i) & " *");
            M.fin_red;
        end loop;
    end redacteur;

    l : lecteur;
    r : redacteur;

begin
    null;
end prio_lecteur;
```

```
tahlisfove@tahlisfove:~/Bureau/tp/rdv_ada/prio_lecteur$ gnatmake prio_lecteur.adb -o prioLecteur
x86_64-linux-gnu-gcc-10 -c prio_lecteur.adb
x86_64-linux-gnu-gnatbind-10 -x prio_lecteur.ali
x86_64-linux-gnu-gnatlink-10 prio_lecteur.ali -o prioLecteur
```

compilation prio_lecteur.adb

```
tahlisfove@tahlisfove:~/Bureau/tp/rdv_ada/prio_lecteur$ ./prioLecteur
debut lect
 * 1 *
fin lect
debut lect
 * 2 *
fin lect
debut lect
 * 3 *
fin lect
debut red
 * 1 *
fin red
debut red
 * 2 *
fin red
debut red
 * 3 *
fin red
```

affichage prioLecteur

a. R0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nbred+1	R0: debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent - rentre en att rédacteur	R1 : debut_red	0	{}	1	{R1}
rédacteur présent - rentre en att lecteur	L1 : debut_lect	0	{L1}	1	{R1}
R1 rentre pas car lecteur attente , L1 à la priorité	R0 : fin_red	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

b. R0 est en cours, L1 → R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun redacteur et lecteur: nbred+1	R0 : debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent – rentre en att lecteur	R1 : debut_red	0	{L1}	1	{}
rédacteur présent – rentre en att redacteur	L1 : debut_lect	0	{L1}	1	{R1}
R1 rentre pas car lecteur L1 à la priorité	R0 : fin_red	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

c. L0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur : nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent - rentre en att rédacteur	R1 : debut_red	1	{}	0	{R1}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

d. L0 est en cours, L1 → R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur : nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{}
lecteur présent - rentre en att rédacteur	R1 : debut_red	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

1.2. Priorité aux rédacteurs

Ici, c'est l'inverse du premier cas, **les rédacteurs ont la priorité sur les lecteurs.** Les **rédacteurs** doivent **attendre que la ressource soit libre** pour écrire, tandis que les **lecteurs peuvent accéder** à la ressource **sans bloquer d'autres lecteurs.** Cette priorité permet d'éviter que les lecteurs occupent constamment la ressource, laissant ainsi l'opportunité aux rédacteurs de travailler.

```
33              -- début lecture
34              when (nbred + debut_red'Count = 0) => accept debut_lect do
```

condition entree lecteur

```
41              -- début écriture
42              when (nblect + nbred = 0) => accept debut_red do
```

condition entree redacteur

Comme pour la priorité lecteur, on les entre (entry) pour chaque cas debut_lect, debut_red, fin_lect et fin_red **dans l'interface Magasinier** ainsi que les conditions entree pour debut_lect et debut_red de sorte a **laisser la priorité** dans notre cas aux **rédacteurs.**

```
with TEXT_IO;
use TEXT_IO;

-- un lecteur et un redacteur avec un tampon de taille N
procedure prio_redacteur is
   package int_io is new Integer_io(integer);
   use int_io;

   -- interface Magasinier
   task type Magasinier is
      entry debut_lect;
      entry fin_lect;
      entry debut_red;
      entry fin_red;
   end Magasinier;

   M : Magasinier;

   -- interface lecteur
   task type lecteur is
   end lecteur;

   -- interface redacteur
   task type redacteur is
   end redacteur;

   -- body Magasinier
   task body Magasinier is

      nblect, nbred, nblectatt : Integer := 0;

   begin
      loop
         select

            -- début lecture
            when (nbred + debut_red'Count = 0) => accept debut_lect do
               nblect := nblect + 1;
               put_line("debut lect");
            end debut_lect;
```

```
        or
           -- début écriture
           when (nblect + nbred = 0) => accept debut_red do
              nbred := nbred + 1;
              put_line("debut red");
           end debut_red;

        or
           -- fin lecture
           accept fin_lect do
              nblect := nblect - 1;
              put_line("fin lect");
           end fin_lect;

        or
           -- fin écriture
           accept fin_red do
              nbred := nbred - 1;
              put_line("fin red");
           end fin_red;

        end select;
     end loop;
  end Magasinier;

  -- corps lecteur
  task body lecteur is
     value : Integer := 1;
  begin
     for i in 1..3 loop
        M.debut_lect;
        put_line(" *" & Integer'Image(i) & " *");
        M.fin_lect;
     end loop;
  end lecteur;

  -- corps redacteur
  task body redacteur is
     value : Integer := 0;
  begin
     for i in 1..3 loop
        M.debut_red;
        put_line(" *" & Integer'Image(i) & " *");
        M.fin_red;
     end loop;
  end redacteur;

  l : lecteur;
  r : redacteur;

begin
  null;
end prio_redacteur;
```

```
tahlisfove@tahlisfove:~/Bureau/tp/rdv_ada/prio_redacteur$ gnatmake prio_redacteur.adb -o prioRedacteur
x86_64-linux-gnu-gcc-10 -c prio_redacteur.adb
x86_64-linux-gnu-gnatbind-10 -x prio_redacteur.ali
x86_64-linux-gnu-gnatlink-10 prio_redacteur.ali -o prioRedacteur
```

compilation prio_redacteur.adb

```
tahlisfove@tahlisfove:~/Bureau/tp/rdv_ada/prio_redacteur$ ./prioRedacteur
debut red
 * 1 *
fin red
debut red
 * 2 *
fin red
debut red
 * 3 *
fin red
debut lect
 * 1 *
fin lect
debut lect
 * 2 *
fin lect
debut lect
 * 3 *
fin lect
```

affichage prioRedacteur

a. R0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nbred+1	R0: debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent - rentre en att rédacteur	R1 : debut_red	0	{}	1	{R1}
rédacteur présent - rentre en att lecteur	L1 : debut_lect	0	{L1}	1	{R1}
R1 rentre car il à la priorité	R0 : fin_red	0	{L1}	1	{}
-	R1 : <ecriture>	0	{L1}	1	{}
plus de redacteur en attente, L1 peut rentrer	R1 : fin_red	1	{}	0	{}
-	L1 : <lecteur>	1	{}	0	{}
-	L1 : fin_lect	0	{}	0	{}

b. R0 est en cours, L1 -> R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nbred+1	R0: debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent - rentre en att lecteur	L1 : debut_lect	0	{L1}	1	{}
rédacteur présent - rentre en att redacteur	R1 : debut_red	0	{L1}	1	{R1}
R1 rentre car il à la priorité	R0 : fin_red	0	{L1}	1	{}
-	R1 : <ecriture>	0	{L1}	1	{}
plus de redacteur en attente, L1 peut rentrer	R1 : fin_red	1	{}	0	{}
-	L1 : <lecteur>	1	{}	0	{}
-	L1 : fin_lect	0	{}	0	{}

c. L0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent - rentre en att rédacteur	R1 : debut_red	1	{}	0	{R1}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	1	{}

d. L0 est en cours, L1 -> R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{}
lecteur présent - rentre en att redacteur	R1 : debut_red	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

1.3. Priorité égales

Dans ce dernier cas, **tous les processus (lecteurs et rédacteurs)** ont un **accès égal** à la ressource **partagée.** Les rendez-vous permettent aux processus d'entrer dès qu'il est prêt, **sans** donner de **priorité spécifique** à un type d'accès. Cela assure une **utilisation équilibrée** de la ressource **sans favoriser les lecteurs ou les rédacteurs.**

```
8   -- Interface  Magasinier
9   task type Magasinier is
10     entry debut_lect;
11     entry fin_lect;
12     entry debut_red;
13     entry fin_red;
14     entry barriere;
15  end Magasinier;
```

interface magasinier

```
35              -- début lecture
36              when (nbred = 0) => accept debut_lect do
37                 bar := True;
38                 nblect := nblect + 1;
39                 put_line("debut lect");
40              end debut_lect;
```

entry debut_lect

```
44          -- début écriture
45          when (nblect + nbred = 0) => accept debut_red do
46             nbred := nbred + 1;
47             put_line("debut red");
48          end debut_red;
```

entry debut_red

Comme pour la priorité lecteur et rédacteur, on les entre pour chaque cas debut_lect, debut_red, fin_lect et fin_red dans l'interface **Magasinier** ainsi que **les conditions entree** pour **debut_lect** et **debut_red** de sorte a laissé **une priorité égale** aux systèmes.

Cependant, on ajoute pour ce dernier cas **une entree barrière** permettant de lorsqu'un redacteur est en cours aucun autre redacteur ne peu s'activer (la barrière est **déclenchée** a l'entré de **debut_red** et **relâché** a la sortie de **fin_red**) les lecteurs eux ne se soucis pas de cette barrière.

```
68          -- barriere
69          when (bar = True) => accept barriere do
70             bar := False;
71          end barriere;
```

entry barriere

```
77    -- corps lecteur
78    task body lecteur is
79       value : Integer := 1;
80    begin
81       for i in 1..3 loop
82          M.barriere;
83          M.debut_lect;
84          put_line(" *" & Integer'Image(i) & " *");
85          M.fin_lect;
86       end loop;
87    end lecteur;
88
89    -- corps redacteur
90    task body redacteur is
91       value : Integer := 0;
92    begin
93       for i in 1..3 loop
94          M.barriere;
95          M.debut_red;
96          put_line(" *" & Integer'Image(i) & " *");
97          M.fin_red;
98       end loop;
99    end redacteur;
```

définition des corps lecteur/redacteur

```
with TEXT_IO;
use TEXT_IO;

-- un lecteur et un redacteur avec un tampon de taille N
procedure prio_egale is
   package int_io is new Integer_io(integer);
   use int_io;

   -- interface Magasinier
   task type Magasinier is
      entry debut_lect;
```

```
   entry fin_lect;
   entry debut_red;
   entry fin_red;
   entry barriere;
end Magasinier;

M : Magasinier;

-- interface lecteur
task type lecteur is
end lecteur;

-- interface redacteur
task type redacteur is
end redacteur;

-- body Magasinier
task body Magasinier is

   nblect, nbred, nblectatt : Integer := 0;
   bar : Boolean := True;

begin
   loop
      select

      -- début lecture
      when (nbred = 0) => accept debut_lect do
         bar := True;
         nblect := nblect + 1;
         put_line("debut lect");
      end debut_lect;

   or
      -- début écriture
      when (nblect + nbred = 0) => accept debut_red do
         nbred := nbred + 1;
         put_line("debut red");
      end debut_red;

   or
      -- fin lecture
      accept fin_lect do
         nblect := nblect - 1;
         put_line("fin lect");
      end fin_lect;
```

```
   or
      -- fin écriture
      accept fin_red do
         nbred := nbred – 1;
         put_line("fin red");
         bar := True;
      end fin_red;

   or
      -- barriere
      when (bar = True) => accept barriere do
            bar := False;
         end barriere;

      end select;
   end loop;
end Magasinier;

-- corps lecteur
task body lecteur is
   value : Integer := 1;
begin
   for i in 1..3 loop
      M.barriere;
      M.debut_lect;
      put_line(" *" & Integer'Image(i) & " *");
      M.fin_lect;
   end loop;
end lecteur;

-- corps redacteur
task body redacteur is
   value : Integer := 0;
begin
   for i in 1..3 loop
      M.barriere;
      M.debut_red;
      put_line(" *" & Integer'Image(i) & " *");
      M.fin_red;
   end loop;
end redacteur;

l : lecteur;
r : redacteur;
```

```
begin
   null;
end prio_egale;
```

```
tahlisfove@tahlisfove:~/Bureau/tp/rdv_ada/prio_egale$ gnatmake prio_egale.adb -o prioEgale
x86_64-linux-gnu-gcc-10 -c prio_egale.adb
x86_64-linux-gnu-gnatbind-10 -x prio_egale.ali
x86_64-linux-gnu-gnatlink-10 prio_egale.ali -o prioEgale
```

compilation prio_egale.adb

```
tahlisfove@tahlisfove:~/Bureau/tp/rdv_ada/prio_egale$ ./prioEgale
debut red
 * 1 *
fin red
debut lect
 * 1 *
fin lect
debut lect
 * 2 *
fin lect
debut lect
 * 3 *
fin lect
debut red
 * 2 *
fin red
debut red
 * 3 *
fin red
```

affichage prioEgale

a. R0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nbred+1	R0: debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent - rentre en att rédacteur	R1 : debut_red	0	{}	1	{R1}
rédacteur présent - rentre en att lecteur	L1 : debut_lect	0	{L1}	1	{R1}
R1 rentre car il à la priorité	R0 : fin_red	0	{L1}	1	{}
-	R1 : <ecriture>	0	{L1}	1	{}
plus de redacteur en attente, L1 peut rentrer	R1 : fin_red	1	{}	0	{}
-	L1 : <lecture>	1	{}	0	{}
-	L1 : fin_lect	1	{}	0	{}

b. R0 est en cours, L1 -> R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nbred+1	R0: debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent - rentre en att lecteur	L1 : debut_lect	0	{L1}	1	{}
rédacteur présent - rentre en att redacteur	R1 : debut_red	0	{L1}	0	{R1}
L1 rentre car il à la priorité	R0 : fin_red	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

c. L0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent - rentre en att rédacteur	R1 : debut_red	1	{}	0	{R1}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	1	{}

d. L0 est en cours, L1 -> R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{}
lecteur présent - rentre en att redacteur	R1 : debut_red	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

Partie 2: Objets/Types protégés ADA 95

1. Principe des Objets/Types protégés en ADA 95

Les **objets et types protégés en Ada 95** sont des **entités** qui assurent un **contrôle** sécurisé des **données partagées** entre les tâches, offrant une **encapsulation des données** et des opérations avec une **gestion synchronisée**.

Leur principe fondamental est **d'assurer l'exclusion mutuelle** et la **synchronisation entre les tâches** via des routines protégées pour **éviter les accès concurrents**, garantissant ainsi la **cohérence des données**. Contrairement aux packages Ada qui organisent le code de manière générale, les objets/types protégés **se concentrent** spécifiquement sur la **sécurité et la cohérence** des accès partagés entre les tâches.

2. Comparaison objets/types protégés aux sémaphores/RDV ADA

Les objets/types protégés offrent une **gestion plus sûre et structurée des données partagées,** avec une **abstraction plus élevée** que les sémaphores et les RDV Ada. Cependant, leur **utilisation** peut être **plus complexe**, nécessitant une compréhension approfondie de la concurrence.

Les **sémaphores et les RDV Ada, moins intuitifs mais plus flexibles** dans certains cas, offrent des mécanismes plus bas-niveau. La décision entre ces outils dépend des **besoins spécifiques du projet** et de la **complexité** de la gestion de la concurrence, tout en considérant la priorité accordée à la sécurité et à la modularité.

3. Implémenter à l'aide Objets/Types protégés

3.1. Producteur/Consommateur

On utilise un objet protégé en Ada pour résoudre un problème de type **producteur-consommateur**. On se sert un objet de type « Magasinier », qui gère **un tampon partagé entre un producteur et un consommateur.** Le producteur utilise l'entrée **produire** pour ajouter des valeurs au tampon, et le consommateur utilise l'entrée **consommer** pour en récupérer. **L'objet protégé assure un accès sécurisé** et ordonné au tampon, **évitant les conflits de données.**

```
12      -- interface Magasinier
13      protected type Magasinier is
14          entry produire (Mess : in Integer);
15          entry consommer (Mess : out Integer);
16      private
17          cpt : Integer := 0;
18          tampon : Tampon_Type;
19          tete, queue : Integer range 0..n-1 := 0;
20      end Magasinier;
```

interface magasinier

```
35      entry produire (Mess : in Integer) when (cpt < n) is
36      begin
37          tampon (tete) := Mess;
38          tete := (tete + 1) mod n;
39          put_line ("produire");
40          cpt := cpt + 1;
41      end produire;
```

entree produire

```
43      entry consommer (Mess : out Integer) when (cpt > 0) is
44      begin
45          Mess := tampon (queue);
46          queue := (queue + 1) mod n;
47          put_line ("consommer");
48          cpt := cpt - 1;
49      end consommer;
```

entree consommer

```
with TEXT_IO;
use TEXT_IO;

-- 1 producteur, 1 consommateur, tampon de taille n
procedure prod_cons_objet_protege is
```

```
package int_io is new Integer_io (integer);
use int_io;

n : constant Integer := 8;
type Tampon_Type is array (0..n-1) of Integer;

-- interface Magasinier
protected type Magasinier is
   entry produire (Mess : in Integer);
   entry consommer (Mess : out Integer);
private
   cpt : Integer := 0;
   tampon : Tampon_Type;
   tete, queue : Integer range 0..n-1 := 0;
end Magasinier;

M : Magasinier;

-- interface Producteur
task type Producteur is
end Producteur;

-- interface Consommateur
task type Consommateur is
end Consommateur;

-- body Magasinier
protected body Magasinier is

   entry produire (Mess : in Integer) when (cpt < n) is
   begin
      tampon (tete) := Mess;
      tete := (tete + 1) mod n;
      put_line ("produire");
      cpt := cpt + 1;
   end produire;

   entry consommer (Mess : out Integer) when (cpt > 0) is
   begin
      Mess := tampon (queue);
      queue := (queue + 1) mod n;
      put_line ("consommer");
      cpt := cpt - 1;
   end consommer;

end Magasinier;
```

```
   -- corps producteur
   task body Producteur is
      value : Integer := 3;
   begin
      for i in 1..3 loop
         M.produire (value);
      end loop;
   end Producteur;

   -- corps consommateur
   task body Consommateur is
      value : Integer := 0;
   begin
      for i in 1..3 loop
         M.consommer (value);
      end loop;
   end Consommateur;

   P : Producteur;
   C : Consommateur;

begin
   null;
end prod_cons_objet_protege;
```

```
tahlisfove@tahlisfove:~/Bureau/tp/objets_proteges_ada/producteur_consommateur$ gnatmake prod_cons_objet_protege.adb -o prodConsOP
x86_64-linux-gnu-gcc-10 -c prod_cons_objet_protege.adb
x86_64-linux-gnu-gnatbind-10 -x prod_cons_objet_protege.ali
x86_64-linux-gnu-gnatlink-10 prod_cons_objet_protege.ali -o prodConsOP
```

compilation producteur_consommateur.adb

```
tahlisfove@tahlisfove:~/Bureau/tp/objets_proteges_ada/producteur_consommateur$ ./prodConsOP
produire
consommer
produire
produire
consommer
consommer
```

affichage prodConsOP

3.2. Lecteurs/Rédacteurs: priorité aux lecteurs et aux rédacteurs

Ici, nous appliquons le **même principe** que celui évoqué lors de la **première partie**. En utilisant des **objets protégés en Ada 95,** les **lecteurs peuvent accéder simultanément à la ressource partagée** sans bloquer d'autres lecteurs. Cependant, leur accès **empêche les rédacteurs d'entrer tant qu'ils lisent,** garantissant une lecture simultanée sans perturber la cohérence des données.

```
9     -- interface  Magasinier
10    protected type Magasinier is
11       entry debut_lect;
12       procedure fin_lect;
13       entry debut_red;
14       procedure fin_red;
15    private
16          nblect, nbred, nblectatt : Integer := 0;
17    end Magasinier;
```

interface Magasinier

```
30          entry debut_lect when (nbred = 0) is
31          begin
```

condition entree debut_lect

```
36       -- début écriture
37       entry debut_red when (nblect + debut_lect'Count = 0) is
38       begin
```

condition entree debut_red

```
with TEXT_IO;
use TEXT_IO;

-- un lecteur et un redacteur avec un tampon de taille N
procedure prio_lecteur_protege is
   package int_io is new Integer_io(integer);
   use int_io;

   -- interface Magasinier
   protected type Magasinier is
      entry debut_lect;
      procedure fin_lect;
      entry debut_red;
      procedure fin_red;
   private
      nblect, nbred, nblectatt : Integer := 0;
   end Magasinier;

   M : Magasinier;

   -- interface lecteur
   task type lecteur is
   end lecteur;

   -- interface redacteur
   task type redacteur is
   end redacteur;

   -- body Magasinier
```

```
protected body Magasinier is

   -- début lecture
   entry debut_lect when (nbred = 0) is
   begin
      nblect := nblect +1;
      put_line ("debut lect");
   end debut_lect;

   -- début écriture
   entry debut_red when (nblect + debut_lect'Count = 0) is
   begin
      nbred := nbred+1;
      put_line ("debut red");
   end debut_red;

   -- fin lecture
   procedure fin_lect is
   begin
      nblect := nblect -1;
      put_line ("fin lect");
   end fin_lect;

   -- fin écriture
   procedure fin_red is
   begin
      nbred:=nbred -1;
      put_line ("fin red");
   end fin_red;

end Magasinier;

-- corps lecteur
task body lecteur is
begin
   for i in 1..3 loop
      M.debut_lect;
      put_line(" *" & Integer'Image(i) & " *");
      M.fin_lect;
   end loop;
end lecteur;

-- corps redacteur
task body redacteur is
begin
   for i in 1..3 loop
```

```
        M.debut_red;
        put_line(" *" & Integer'Image(i) & " *");
        M.fin_red;
     end loop;
  end redacteur;

  l : lecteur;
  r : redacteur;

begin
  null;
end prio_lecteur_protege;
```

```
tahlisfove@tahlisfove:~/Bureau/tp/objets_proteges_ada/prio_lecteur$ gnatmake prio_lecteur_protege.adb -o prioLP
x86_64-linux-gnu-gcc-10 -c prio_lecteur_protege.adb
x86_64-linux-gnu-gnatbind-10 -x prio_lecteur_protege.ali
x86_64-linux-gnu-gnatlink-10 prio_lecteur_protege.ali -o prioLP
```

compilation prio_lecteur_protege.adb

```
tahlisfove@tahlisfove:~/Bureau/tp/objets_proteges_ada/prio_lecteur$ ./prioLP
debut lect
 * 1 *
fin lect
debut lect
 * 2 *
fin lect
debut lect
 * 3 *
fin lect
debut red
 * 1 *
fin red
debut red
 * 2 *
fin red
debut red
 * 3 *
fin red
```

affichage prioLP

a. R0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nbred+1	R0: debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent - rentre en att rédacteur	R1 : debut_red	0	{}	1	{R1}
rédacteur présent - rentre en att lecteur	L1 : debut_lect	0	{L1}	1	{R1}
R1 rentre pas car lecteur attente, L1 à la priorité	R0 : fin_red	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

b. R0 est en cours, L1 → R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun redacteur et lecteur: nbred+1	R0 : debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent – rentre en att lecteur	R1 : debut_red	0	{L1}	1	{}
rédacteur présent – rentre en att redacteur	L1 : debut_lect	0	{L1}	1	{R1}
R1 rentre pas car lecteur L1 à la priorité	R0 : fin_red	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

c. L0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur : nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent - rentre en att rédacteur	R1 : debut_red	1	{}	0	{R1}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

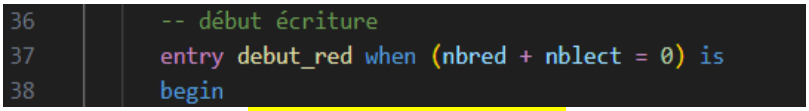
d. L0 est en cours, L1 → R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur : nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{}
lecteur présent - rentre en att rédacteur	R1 : debut_red	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur en attente, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

De même **pour les rédacteurs** qui ont la priorité sur les lecteurs. Lorsqu'un rédacteur modifie la ressource partagée, il **bloque l'accès pour tous les autres,** lecteurs ou rédacteurs. Cette priorité assure **l'intégrité des données pendant les modifications**, même si cela peut **temporairement empêcher l'accès** à la ressource pour d'autres processus.

```
29        -- début lecture
30        entry debut_lect when (nbred + debut_red'Count = 0) is
31        begin
```

condition entree debut_lect



```
36      -- début écriture
37      entry debut_red when (nbred + nblect = 0) is
38      begin
```

condition entree debut_red

```
with TEXT_IO;
use TEXT_IO;

-- un lecteur et un redacteur avec un tampon de taille N
procedure prio_redacteur_protege is
   package int_io is new Integer_io(integer);
   use int_io;

   -- interface Magasinier

   protected type Magasinier is
      entry debut_lect;
      procedure fin_lect;
      entry debut_red;
      procedure fin_red;
   private
      nblect, nbred, nblectatt, nbredatt : Integer := 0;
   end Magasinier;

   M : Magasinier;

   -- interface lecteur
   task type lecteur is end lecteur;

   -- interface redacteur
   task type redacteur is end redacteur;

   -- body Magasinier
   protected body Magasinier is

      -- début lecture
      entry debut_lect when (nbred + debut_red'Count = 0) is
      begin
         nblect:=nblect +1;
         put_line ("debut lect");
      end debut_lect;

      -- début écriture
      entry debut_red when (nbred + nblect = 0) is
      begin
         nbred:=nbred+1;
```

```
        put_line ("debut red");
    end debut_red;

    -- fin lecture
    procedure fin_lect is
    begin
        nblect:=nblect -1;
        put_line ("fin lect");
    end fin_lect;

    -- fin écriture
    procedure fin_red is
    begin
        nbred:=nbred -1;
        put_line ("fin red");
    end fin_red;

end Magasinier;

-- corps lecteur
task body lecteur is
begin
    for i in 1..3 loop
        M.debut_lect;
        put_line(" *" & Integer'Image(i) & " *");
        M.fin_lect;
    end loop;
end lecteur;

-- corps redacteur
task body redacteur is
begin
    for i in 1..3 loop
        M.debut_red;
        put_line(" *" & Integer'Image(i) & " *");
        M.fin_red;
    end loop;
end redacteur;

l : lecteur;
r : redacteur;

begin
   null;
end prio_redacteur_protege;
```

```
tahlisfove@tahlisfove:~/Bureau/tp/objets_proteges_ada/prio_redacteur$ gnatmake prio_redacteur_protege.adb -o prioRP
x86_64-linux-gnu-gcc-10 -c prio_redacteur_protege.adb
x86_64-linux-gnu-gnatbind-10 -x prio_redacteur_protege.ali
x86_64-linux-gnu-gnatlink-10 prio_redacteur_protege.ali -o prioRP
```

compilation prio_redacteur_protege.adb

```
tahlisfove@tahlisfove:~/Bureau/tp/objets_proteges_ada/prio_redacteur$ ./prioRP
debut red
 * 1 *
fin red
debut red
 * 2 *
fin red
debut red
 * 3 *
fin red
debut lect
 * 1 *
fin lect
debut lect
 * 2 *
fin lect
debut lect
 * 3 *
fin lect
```

affichage prioRP

e. R0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nbred+1	R0: debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent - rentre en att rédacteur	R1 : debut_red	0	{}	1	{R1}
rédacteur présent - rentre en att lecteur	L1 : debut_lect	0	{L1}	1	{R1}
R1 rentre car il à la priorité	R0 : fin_red	0	{L1}	1	{}
-	R1 : <ecriture>	0	{L1}	1	{}
plus de redacteur en attente, L1 peut rentrer	R1 : fin_red	1	{}	0	{}
-	L1 : <lecteur>	1	{}	0	{}
-	L1 : fin_lect	0	{}	0	{}

f. R0 est en cours, L1 -> R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nbred+1	R0: debut_red	0	{}	1	{}
-	R0 : <ecriture>	0	{}	1	{}
rédacteur présent - rentre en att lecteur	L1 : debut_lect	0	{L1}	1	{}
rédacteur présent - rentre en att redacteur	R1 : debut_red	0	{L1}	1	{R1}
R1 rentre car il à la priorité	R0 : fin_red	0	{L1}	1	{}
-	R1 : <ecriture>	0	{L1}	1	{}
plus de redacteur en attente, L1 peut rentrer	R1 : fin_red	1	{}	0	{}
-	L1 : <lecteur>	1	{}	0	{}
-	L1 : fin_lect	0	{}	0	{}

g. L0 est en cours, R1 → L1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent - rentre en att rédacteur	R1 : debut_red	1	{}	0	{R1}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	1	{}

h. L0 est en cours, L1 -> R1

commentaires	étapes	nbLect	lectAtt	nbRed	redAtt
-	-	0	{}	0	{}
aucun lecteur et redacteur: nblect+1	L0: debut_lect	1	{}	0	{}
-	L0 : <lecture>	1	{}	0	{}
lecteur présent: nblect+1	L1 : debut_lect	2	{}	0	{}
lecteur présent - rentre en att redacteur	R1 : debut_red	2	{}	0	{R1}
R1 rentre pas car lecteur L1 à la priorité	L0 : fin_lect	1	{}	0	{R1}
-	L1 : <lecture>	1	{}	0	{R1}
plus de lecteur, R1 peut rentrer	L1 : fin_lect	0	{}	1	{}
-	R1 : <ecriture>	0	{}	1	{}
-	R1 : fin_red	0	{}	0	{}

3.3. Le problème du Carrefour a sens giratoire

Dans ce cas, on simule le **fonctionnement d'un rond-point** pour la circulation routière en utilisant un **objet protégé nommé Giratoire.** Cet objet contrôle les **entrées** et **sorties** des voitures dans le rond-point **pour éviter les collisions.**

L'entrée entree permet aune voiture **d'entrée depuis une voie spécifique,** tandis que la procédure sortir **gère la sortie d'une voiture du rond-**point et dans le cas ou il n'y a plus de voiture présente la recherche de la prochaine **voie prioritaire.** L'objet protégé assure **un accès** ordonné, **une voie** à la fois et un max de 20 voitures simultanées, pour **maintenir la sécurité/fluidité du trafic.**

```
16    -- interface carrefour
17    protected type Giratoire is
18       -- signal d'entree dans le carrefour
19       entry entree(Typevoie);
20       -- signal de sortie du carrefour
21       procedure sortir;
22    private
23       CAP_MAX : Integer := 20;
24       nbvoiture : Integer := 0;
25       VoieCourante : Integer := 0;
26       CarrefourVide : Boolean := true;
27       i : Integer := 0;
28    end Giratoire;
```

interface Giratoire

```
35      -- entrée giratoire
36      entry entree(for voie in Typevoie) when (voie = VoieCourante and nbvoiture < CAP_MAX) is
37      begin
38         nbvoiture := nbvoiture + 1;
39         VoieCourante := voie;
40         CarrefourVide := false;
41         put_line(" * Une voiture entre voie" & Integer'Image(voie) & " *");
42      end entree;
```

procedure entree

```
44      -- sortir giratoire
45      procedure sortir is
46      begin
47         nbvoiture := nbvoiture - 1;
48         if (nbvoiture = 0) then
49            i := VoieCourante + 1 mod NbVoie;
50            VoieCourante := -1;
51
52            -- recherche de la prochaine voie courante
53            while entree(i)'Count = 0 loop
54               i := i + 1 mod NbVoie;
55            end loop;
56               VoieCourante := i;
57               CarrefourVide := true;
58         end if;
59      end sortir;
```

procedure sortir

```
with TEXT_IO;
use TEXT_IO;

procedure giratoire is
   package int_io is new Integer_io(integer);
   use int_io;

   -- nombre de voies dans le giratoire
   NbVoie : Integer := 5;

   -- interface Typevoie
   Subtype Typevoie is Integer range 0..NbVoie-1;

   -- interface voiture
   task type voiture(v:Typevoie) is
   end voiture;

   -- interface carrefour
   protected type Giratoire is
      -- signal d'entree dans le carrefour
      entry entree(Typevoie);
      -- signal de sortie du carrefour
      procedure sortir;
   private
      CAP_MAX : Integer := 20;
```

```
   nbvoiture : Integer := 0;
   VoieCourante : Integer := 0;
   CarrefourVide : Boolean := true;
   i : Integer := 0;
end Giratoire;

G : Giratoire;

-- body Giratoire
protected body Giratoire is

   -- entrée giratoire
   entry entree(for voie in Typevoie) when (voie = VoieCourante and nbvoiture < CAP_MAX) is
   begin
      nbvoiture := nbvoiture + 1;
      VoieCourante := voie;
      CarrefourVide := false;
      put_line(" * Une voiture entre voie" & Integer'Image(voie) & " *");
   end entree;

   -- sortir giratoire
   procedure sortir is
   begin
      nbvoiture := nbvoiture - 1;
      if (nbvoiture = 0) then
         i := VoieCourante + 1 mod NbVoie;
         VoieCourante := -1;

         -- recherche de la prochaine voie courante
         while entree(i)'Count = 0 loop
            i := i + 1 mod NbVoie;
         end loop;
         VoieCourante := i;
         CarrefourVide := true;
      end if;
   end sortir;

end Giratoire;

-- corps voiture
task body voiture is
begin
   put_line("Voiture attente voie" & Integer'Image(v));
   delay 1.0;
   G.entree(v);
   put_line("sortie du giratoire");
   G.sortir;
end voiture;

v1: voiture(4);
v2: voiture(2);
v3: voiture(3);
v4: voiture(1);
```

```
   v5: voiture(3);
   v6: voiture(0);
   v7: voiture(3);
   v8: voiture(1);

begin
   null;
end giratoire;
```

```
tahlisfove@tahlisfove:~/Bureau/tp/objets_proteges_ada/giratoire$ gnatmake giratoire.adb -o giratoire
x86_64-linux-gnu-gcc-10 -c giratoire.adb
x86_64-linux-gnu-gnatbind-10 -x giratoire.ali
x86_64-linux-gnu-gnatlink-10 giratoire.ali -o giratoire
```

compilation giratoire.adb

```
tahlisfove@tahlisfove:~/Bureau/tp/objets_proteges_ada/giratoire$ ./giratoire
Voiture attente voie 1
Voiture attente voie 1
Voiture attente voie 3
Voiture attente voie 4
Voiture attente voie 2
Voiture attente voie 3
Voiture attente voie 0
Voiture attente voie 3
 * Une voiture entre voie 0 *
sortie du giratoire
 * Une voiture entre voie 1 *
 * Une voiture entre voie 1 *
sortie du giratoire
sortie du giratoire
 * Une voiture entre voie 2 *
sortie du giratoire
 * Une voiture entre voie 3 *
 * Une voiture entre voie 3 *
 * Une voiture entre voie 3 *
sortie du giratoire
sortie du giratoire
sortie du giratoire
 * Une voiture entre voie 4 *
sortie du giratoire
```

affichage giratoire