Communication et synchronisation en Ada
Les objets Protégés

Samia Bouzefrane

Maître de Conférences

CEDRIC –CNAM

samia.bouzefrane@cnam.fr
http://cedric.cnam.fr/~bouzefra

Sommaire

- **Synchronisation en Ada**
  - Présentation des objets protégés d'Ada
  - Sémantique de base
  - Exemple d'utilisation

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Synchronisation en ADA

Dans le modèle de synchronisation proposé par Ada :

- **Il n'existe pas de sémaphores**

- **Deux mécanismes sont proposés :**

– Les « objets protégés » : proche des moniteurs; permettent l'encapsulation de données privées qui sont accédées par des fonctions, des procédures ou des entrées.

– L'utilisation de « rendez-vous » : proche des RPC entre une tâche serveur qui accepte le rendez-vous et une tâche client qui appelle le serveur pour ce rendezvous

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé

➢**Rappel** : il faut distinguer deux sortes d'objets :
- objets *actifs* comme les processus et les tâches
- objets *passifs* comme les moniteurs, les ressources, les objets protégés

➢**Un objet protégé**
- est un objet partageable

- C'est un élément « passif » : ce sont les tâches qui exécutent le code des sous-programmes ou entrées des objets protégés

- définit des données privées ne pouvant être accédées que par les sous-programmes (fonctions, procédures ou entrées) associés à l'objet protégé

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé

- Chaque accès (fonctions, procédures et entrées) est une section critique protégée par un verrou; il y a un verrou par objet protégé; s'il y a plusieurs appels concurrents à un objet protégé, un seul est pris, les autres sont en "attente externe",

- Syntaxe à respecter :

protected *NomDeLObjetProtege* ***is***
déclaration des fonctions, procédures et entrées de l'objet protégé
private
déclaration des données privées de l'objet protégé
end *NomDeLObjetProtege****;***

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: sémantique de base

La partie privée (« **`private`** ») définit les données ou contenu de l'objet protégé :

➢ Les fonctions définissent des actions de « lecture » du contenu de l'objet protégé (interdiction de modifier la valeur de celles-ci)

➢ Les procédures et les entrées définissent des actions de « lecture » et «d'écriture » des données de l'objet protégé

➢ Plusieurs lectures peuvent avoir lieu simultanément (si le compilateur implante un schéma lecteurs -rédacteurs)

➢ Une action d'écriture exclut toute autre action (lecture ou écriture)

➢ Contrairement à une procédure ou à une fonction, l'appel à une entrée peut être bloquant.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Les entrées d'un objet protégé

Une entrée permet de définir des traitements sous conditions

– Chaque entrée possède une file d'attente à laquelle est associée une expression booléenne que l'on appelle « garde » ou « barrière ». Dans ce cas on utilise la clause :

```
entry e when expression_booléenne is...
```

– L'entrée n'est « ouverte » que si cette expression (la garde) est vraie

– Lorsque l'entrée est « fermée » (la garde est fausse), les appels sur cette entrée sont mis en attente (ils ne seront traités que lorsque la garde redeviendra vraie)

– Une tâche, exécutant le code d'une entrée, peut être placée dans la file d'attente d'une autre entrée par l'instruction « requeue »

– Une tâche en attente sur une entrée interne est plus prioritaire qu'une tâche faisant un "nouvel" appel à une entrée (ou procédure) de l'objet protégé.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Les entrées d'un objet protégé

➢ **Si un appel est fait sur une entrée, alors la garde est évaluée;**

- l'évaluation de la garde est en section critique;
- si la garde est vraie, l'entrée est exécutée en exclusion mutuelle;
- si la garde est fausse, la tâche appelante est mise en "attente interne" et est placée dans une file d'attente associée à l'entrée appelée;

➢ **Quand une procédure ou une entrée de l'objet protégé se termine, toutes les gardes des entrées qui ont une tâche en attente interne sont évaluées;**

- si une garde au moins est vraie, l'une des tâches en attente interne sur cette entrée (et une seule) est choisie et l'accès exclusif à l'objet lui est attribué;
- le choix d'une tâche parmi plusieurs possibles est non déterministe;
- les tâches en attente interne ont priorité sur les tâches en attente externe; en particulier, un nouvel appel d'entrée ne peut pas évaluer de garde tant que l'appel en cours n'est pas terminé

➢ **A la fin d'une fonction, les gardes ne sont pas réévaluées puisque la fonction ne modifie pas de valeur.**

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: syntaxe/1

Déclaration d'un objet protégé avec des entrées :

```
protected type obj_protege is
                entry A(..);
                entry B(..);
```

Définition du corps de l'objet protégé :

```
protected body obj_protege is
        entry A (..) is
        begin ..
        end A;

        entry B (..) is
        begin ..
        end B;
end obj_protege;
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: syntaxe/2

Déclaration d'une variable de type objet protégé :

```
Obj : obj_protege;
```

Appel des entrées de l'objet protégé dans un programme :

```
Obj.A(..);
...
Obj.B(..);
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: Exemple

Écrire un programme qui crée deux tâches qui partagent un buffer (à une case) de telle sorte qu'une tâche est productrice d'un message et que l'autre en est la consommatrice. Par exemple, une tâche génère un nombre et le dépose dans le buffer et l'autre récupère la valeur contenue dans le buffer en vue de l'afficher.

Le buffer sera implanté à l'aide d'un objet protégé.

Définir, dans un paquetage de données, un objet protégé à deux points d'entrée :
`-Déposer (var : in integer)` appelée par la tâche productrice
`-Retirer (var : out integer)` appelée par la tâche consommatrice.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: Prod/Cons

```
-- fichier: objet_buffer.ads


package objet_buffer is

    protected type Buffer is
        entry Deposer(X: in float);
        entry Retirer(X: out float);
        private
                Donnee : float;
                Plein : boolean := False;
    end Buffer;

end objet_buffer;
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: Prod/Cons

```
-- fichier : objet_buffer.adb
package body objet_buffer is

protected body Buffer is
    entry Deposer (X: in float) when not Plein is
    begin
        Donnee :=X;
        Plein := True;
    end Deposer;

    entry Retirer (X: out float) when Plein is
    begin
        X:= Donnee;
        Plein := False;
    end Retirer;
end Buffer;

end objet_buffer;
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: Prod/Cons

```
-- fichier : prod_cons.adb
with text_io; use text_io;
with objet_buffer; use objet_buffer;
with ada.numerics.float_random; use ada.numerics.float_random;


procedure prod_cons is

package es_reel is new text_io.float_io(float);
use es_reel;

N: constant :=3;
Buf: Buffer;


task type Producteur;
task type Consommateur;
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: Prod/Cons

```
-- suite de prod_cons.adb
task body Producteur is
G: generator;
mess : float;
        begin
                reset(G);
                mess := Random(G);
                -- produire une valeur
                Buf.Deposer(mess);
                put("Producteur "); put(mess);
                put(" depose ");
                put(mess);
                put_line("--fin Prod");
end Producteur;

task body Consommateur is
c: Float;
        begin
                Buf.Retirer(c);
                put ("Consommateur");
                put(" vient de lire ");
                put(c);
                put_line("--fin Cons");
end Consommateur;
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Objet protégé: Prod/Cons

```
-- fin de prod_cons.adb
Prod : array (1..N) of Producteur;
Cons: array (1..N) of Consommateur;

begin
        null;
end prod_cons;
```

Exécution:

```
$gcc objet_buffer.adb

$gcc prod_cons.adb

$./prod_cons
```

…

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Conclusion

➢**Ada n'offre pas de sémaphore mais propose :**

– les objets protégés qui ont la sémantique des moniteurs et qui se manipulent très simplement

– offre aussi le concept de Rendez-Vous inexistant dans d'autres langages.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Références

Samia Bouzefrane, supports de cours sur le Temps Réel Asynchrone (NFP227) :
http://cedric.cnam.fr/~bouzefra

Claude Kaiser, supports de cours ACCOV (NFP103) :
http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/ACCOV/

Jean-François Peyre, supports de cours sur l'informatique industrielle-systèmes temps réel, CNAM(Paris).