Communication et synchronisation en Ada
Le concept de Rendez-Vous

Samia Bouzefrane

Maître de Conférences

CEDRIC –CNAM

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

Sommaire

- **Synchronisation en Ada**
  - Présentation du concept de Rendez-Vous en 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) -

Le Rendez-Vous entre tâches Ada

- **Rendez-vous asymétrique entre des clients appelants et un serveur appelé**
  - le serveur est connu des clients;
  - Mais le serveur ne connaît pas les clients qui l'appellent

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

Le Rendez-Vous Ada : Côté Serveur/1

- **La tâche serveur possède des points d'appel (entrées) déclarées par entry**
- **La tâche serveur accepte le rendez-vous avec l'instruction accept**
- **C'est un rendez-vous étendu : appel + traitement + retour**
- **Il y a passage de paramètres données *in* et résultats *out***
- **L'acceptation du rendez-vous se fait sous condition (garde) :**

```
when condition => accept
```

la condition ne peut contenir ni le nom ni les paramètres du client

- **Fin du rendez-vous à l'initiative du serveur : accept ... do ... end;**

do...end permet au serveur de faire des actions avec un client bloqué
A la fin du rendez-vous, les résultats sont renvoyés au client qui est réveillé

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

Le Rendez-Vous Ada : Côté Serveur/2

➢ **Le rendez-vous peut être traité par plusieurs entrées successivement avec l'instruction requeue sans libérer l'appelant au moment du requeue**

➢**Attente sélective (1 parmi P choix possibles) : clause select**

unicité de l'accept : un seul rendez-vous quel que soit le nombre d'appels consultés par select

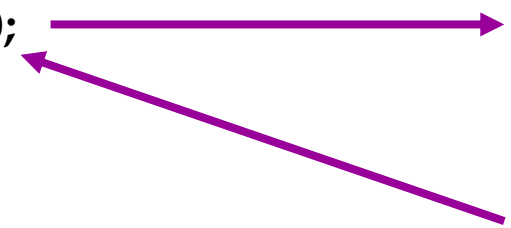
Il y a indéterminisme : rendez-vous avec l'un quelconque des possibles

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

Sémantique de l'invocation à distance: Rendez-vous étendu

CLIENT APPELANT

d : demande; r : réponse; ...
d := calcule_la_demande;
serveur.service(d,r);



SERVEUR APPELÉ

accept service(x: in demande;
y : out réponse)
do y := f(x);
end service;

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

Exemple d'utilisation/1

La tâche *Serveur* définit un point de RDV *proc_rdv()*:

```
--spécification
task Serveur is
      entry proc_rdv(...);
end Serveur;
```


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

Exemple d'utilisation/2

La tâche *Serveur* définit aussi dans son corps la procédure (le service) à exécuter lors du RDV :

```
-- acceptation du RDV
task body Serveur is
....
begin
      instructions;
      accept proc_rdv(...) do -- définit des paramètres
      -- formels comme dans une procédure
            instructions;
      end proc_rdv;
      instructions;
end Serveur ;
```


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

Exemple d'utilisation /3

La tâche *Client* fait appel à la procédure de RDV de la tâche Serveur :

```
-- la tâche Client qui sera en RDV avec la tâche Serveur
task body Client is
....
begin
      instructions;
      Serveur.proc_rdv(...); -- demande de RDV
      instructions;
end Client;
```

L'appel d'un RDV permet en général de communiquer des données produites par la tâche cliente et récupérées par la tâche serveur qui offre le RDV.

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

Exemple simple

```
with TEXT_IO; use TEXT_IO; -- paquetage standard d'impression
procedure PERE is -- création du père, programme principal
        task type TT is -- déclaration de l'interface du type fils
                entry PRINT( ID : INTEGER); -- interface de RDV
        end TT;
        FILS1, FILS2 : TT; -- création de deux fils serveurs

        task body TT is -- déclaration du corps des fils
        begin
                accept PRINT( ID : INTEGER) do
                        PUT_LINE ("BONJOUR DU FILS "& INTEGER'IMAGE(ID));
                end PRINT;
        end TT;
begin -- début du père (client) et activation des fils serveurs
        PUT_LINE ("BONJOUR DU PERE");
        FILS1.PRINT(1); -- appel et réveil du fils1 avec une requête
        FILS2.PRINT(2); -- appel et réveil du fils2 avec une requête
        PUT_LINE ("FIN DU PROGRAMME");
end PERE;
```


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

Autre exemple

Écrire un programme qui crée deux tâches qui doivent se synchroniser par rendez-vous en vue de convertir des caractères. La tâche cliente produit un caractère et demande à la tâche serveur de convertir son caractère en majuscule. Une fois le caractère converti, il est retourné à la tâche cliente qui l'affiche.

Le point de RDV est représenté à l'aide de la procédure **`service`***()*.

La tâche serveur doit définir un point d'entrée avec le nom de cette procédure.

```
entry service(min: in character, maj: out character);
```


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

Autre exemple (suite)

```
with text_io; use text_io;
procedure serviceMaj is
        -- rdv du client et du serveur
        -- en vue de produire un caractère converti en majuscule

        function majuscule (c: character) return character is
        begin
        if c>='a' and c<='z' then
                return
character'val(character'pos(c)-character'pos('a')+character'pos('A'));
           else return c;
        end if;
        end majuscule;
```


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

Le Rendez-Vous en ADA: exemple/3

```
-- spécification des taches client et serveur
        task  client;
        task  serveur is
                -- vue externe de la tache
                entry service(min: in character;maj: out character );
                -- 'service' est un point de synchronisation
                -- c est un parametre formel comme dans un sous-pgme
        end serveur;

        -- corps de client
        task body client is
                c1,c2: character;
                begin
                   while not end_of_file(standard_input) loop
                        get(c1);
                        serveur.service(c1,c2);
                        put(c2);
                    end loop;
        end client;
```


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

Le Rendez-Vous en ADA: exemple/4

```
-- corps de serveur
task body serveur is
        begin
           loop
                accept service(min: in character;maj: out character ) do
                -- les noms des taches appelantes ne sont pas spécifiées
                maj:=majuscule(min);
                end service;
            end loop;
end serveur;

begin
        null;
end serviceMaj;
```


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

Conclusion

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

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

– offre aussi le concept de Rendez-Vous (proche du modèle Client/Serveur).

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).

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