CENTRALE
L Y O N

Le Multi - Tâche en ADA

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Langage ADA: Tâches

- **Dans des programmes séquentiels les instructions sont suivies dans l'ordre.**
- **Dans beaucoup d'applications, il est utile d'écrire un programme sous forme de plusieurs activités parallèles qui coopèrent selon les besoin.**
- **En ADA, les activités parallèles sont décrites par l'intermédiaire de tâches.**
- **Dans les cas simples, une tâche est décrite lexicalement par une forme proche de celle d'un paquetage (spécification et corps):**

```
task T is              -- spécification
   .....
end T;
task body T is         -- corps
   ......
end T;
```

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Exemple : Courses d'une famille (1)

ACHETER_VIANDE, ACHETER_SALADE et ACHETER_VIN.

```
procedure COURSES is
begin
        ACHETER_VIANDE;
        ACHETER_SALADE;
        ACHETER_VIN;
  end;
```

Solution séquentielle, pas très efficace car on n'utilise qu'un seul processeur.

BTD/GL-ADA

3

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Exemple : Courses d'une famille (2)

Solution parallèle

```
Procedure COURSES is

task PRENDRE_SALADE;
task body PRENDRE_SALADE is
begin
     ACHETER_SALADE;
end PRENDRE_SALADE;

task PRENDRE_VIN;
task body PRENDRE_VIN is
begin
     ACHETER_VIN;
end PRENDRE_VIN;

begin
     ACHETER_VIANDE;
end COURSES;
```

- **Avec : mère comme processeur central (elle s'occupe de la VIANDE) et les processeurs secondaires (SALADE et VIN)**

BTD/GL-ADA

4

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Principes de vie d'une tâche

- L'activation d'une tâche est automatique : les tâches locales deviennent actives quand l'unité mère atteint le BEGIN qui suit les déclarations de tâches.
- Une telle tâche se termine quand elle atteint le END final.
- On dit d'une tâche déclarée dans la partie déclarative d'un sous-programme, d'un bloc ou d'un corps de tâche qu'elle dépend de ce cadre.
- Une règle importante stipule qu'on ne peut quitter ce cadre avant que toutes les tâches dépendantes soient terminées.
- La terminaison comporte deux étapes:
  - nous disons qu'un cadre est achevé lorsqu'il atteint son end final.
  - il deviendra par la suite terminé seulement quand toutes les tâches dépendantes, s'il y en a, seront aussi terminées.

BTD/GL-ADA

5

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Le rendez-vous

- En général les tâches interagissent entre elles. En ADA la réalisation passe par un mécanisme appelé rendez-vous.
- Un rendez-vous entre deux tâches a lieu en conséquence de l'appel de l'entrée d'une tâche par une autre tâche. Une entrée est déclarée dans la partie spécification de la tâche:

```
task T is
    entry E(...);
end;
```

- Une entrée peut avoir des paramètres de mode in, out, in-out, mais pas de résultat comme une fonction.

On appelle une entrée comme une procédure :

```
T.E(...);
```

Un nom de tâche ne peut figurer dans une clause use (la notation pointée est nécessaire pour appeler l'entrée depuis l'extérieur de la tâche). Une tâche locale peut appeler une entrée de sa mère directement (règles usuelles de portée et de visibilité)

BTD/GL-ADA

6

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Prise en compte du rendez-vous

Le déroulement des instructions à effectuer lors d'un rendez-vous est décrit par les instructions ACCEPT se trouvant dans le corps de la tâche contenant la déclaration de l'entrée.

```
accept E(...) do
-- suite d'instructions
end E;
```

Les paramètres formels de l'entrée E sont répétés.

Le corps de l'instruction ACCEPT ne contient qu'une suite d'instructions. (Tout traite-exception ou déclaration doit être fourni dans le bloc local.)

BTD/GL-ADA 7

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Procédure vs Tâche

- **La différence la plus importante entre un appel d'entrée et un appel de procédure est la suivante:**
  - **Procédure : exécution immédiate par le processeur en charge de l'exécution du programme**
  - **tâche: une tâche appelle l'entrée mais c'est la tâche à laquelle appartient l'entrée qui exécute l'instruction ACCEPT correspondante, car chaque tâche est gérée par un processeur.**
- **De plus, l'instruction ACCEPT n'est exécutée que lorsqu'une tâche appelle l'entrée et que la tâche possédant l'entrée atteint l'instruction ACCEPT (la tâche en avance est suspendue).**

BTD/GL-ADA 8

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Exemple de tampon actif une place

```
task TAMPON_ACTIF is
        entryDEPOSER (X: in ITEM);
        entryPRENDRE (X: out ITEM);
end;

task body TAMPON_ACTIF is
        V : ITEM;
        Begin
        Loop
                accept DEPOSER (X: in ITEM) do
                        V := X;
                end DEPOSER;
                accept PRENDRE (X: out ITEM) do
                        X := V;
                end PRENDRE;
        end loop;
end TAMPON_ACTIF;
```

- Les autres tâches peuvent alors donner au acquerir des items en appelant :

```
TAMPON_ACTIF.DEPOSER (...);

TAMPON_ACTIF.PRENDRE (...);
```

Boucle infinie, alternance entre DEPOSER et PRENDRE, mise en file d'attente de type FIFO (premier venu premier servi).

BTD/GL-ADA

9

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Fonctionnement

- Dans le corps de la tâche possédant l'entrée E on dispose de E'COUNT qui fournit le nombre de tâches dans la file d'attente de l'entrée E.
- Chaque exécution d'une instruction ACCEPT retire une tâche de la file d'attente.
- Notez l'asymétrie du rendez-vous. La tâche appelante doit nommer la tâche appelée mais pas vice-versa.
- Plusieurs tâches peuvent appeler une entrée et être mises en file d'attente, mais une tâche ne peut être que dans une file à la fois.
- Les entrées peuvent se surcharger les unes les autres et se surcharger avec des sous-programmes (pour éviter la notation pointée).
- Synchronisation simple :

Déclaration :	entry SIGNAL;
Appel :	T.SIGNAL;
accept sans corps :	accept SIGNAL;

Si une tâche appelle l'une de ses propres entrées cela conduira automatiquement à un blocage.

BTD/GL-ADA

10

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Temps et ordonnancement

- Conceptuellement chaque tâche a son processeur, dans un contexte réel, il faut un algorithme d'ordonnancement :
  - Priorité : pragma PRIORITY (7), priorité est une expression statique du sous-type PRIORITY du type INTEGER, on précise l'éligibilité d'une tâche
  - Délai : delay 3.0; attendre au moins 3 secondes

BTD/GL-ADA

11

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Instruction select

- L'instruction SELECT permet à une tâche de choisir un rendez-vous parmi plusieurs possibles.
- Plusieurs possibilités sont offertes :
  - Select simple
  - Select avec condition de garde
  - Select avec delay
  - Select avec else

BTD/GL-ADA

12

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Select simple

```
package VARIABLE_PROTEGEE is
      procedure LIRE (X : out ITEM);
      procedure ECRIRE (X : in ITEM);
end;

package body VARIABLE_PROTEGEE is
      V : ITEM;
      procedure LIRE (X : out ITEM) is
            begin
                  X :=V;
            end;
      procedure ECRIRE (X : in ITEM) is
            begin
                  V := X;
            end;
Begin
      v := VALEUR_INITIALE;
end VARIABLR_PROTEGEE;

Type ITEM is
      record
            X_COORD : REAL;
            Y_COORD : REAL;
      end record;
```

```
task VARIABLE_PROTEGEE is
      entryLIRE (X : out ITEM);
      entryECRIRE (X : in ITEM);
end;

task body VARIABLE_PROTEGEE is
      V : ITEM;
begin
      accept ECRIRE (X : in ITEM) do
            V := X;
      end;
      loop
            select
               accept LIRE (X : out ITEM) do
                     X := V;
               end;
            or
               accept ECRIRE (X : in ITEM) do
                     V := X;
               end;
            end select,
      end loop;
End VARIABLE_PROTEGEE;
```

BTD/GL-ADA

13

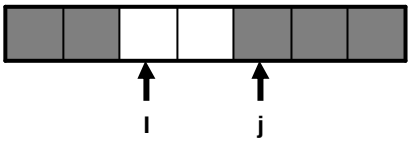
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Select avec condition de garde (when condition =>)

```
task TAMPON_ACTIF is
      entry DEPOSER (X : out ITEM);
      entry PRENDRE (X : in ITEM);
end;

task body TAMPON_ACTIF is
      N : constant := 8; -- Par exemple
      A : array (1..N) of ITEM;
      I, J : INTEGER range 1::N :=1;
      DECOMPTE : INTEGER range 0::N := 0;
begin
      loop
        select
            when DECOMPTE < N =>
            accept DEPOSER (X : in ITEM) do
                  A(I) := X;
            end;
            I:= I mod N+1; DECOMPTE:=DECOMPTE+1;
      or
            when DECOMPTE>0 =>
            accept PRENDRE (X : out ITEM) do
                  X := A(J);
             end;
             J:=J mod N+1; DECOMPTE:=DECOMPTE -1,
             end;
        end select,
      end loop;
end TAMPON_ACTIF;
```



BTD/GL-ADA

14

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Exemple Lecteurs-Ecrivain

```
package LECTEURS_ECRIVAIN is
      procedure LIRE (X : out ITEM);
      procedure ECRIRE (X : in ITEM);
end;

 package body LECTEURS_ECRIVAIN is
      V : ITEM;
      task CONTROLE is
                entryDEBUT;
                entryFIN;
                entryECRIRE (X : in ITEM);
      end;
 task body CONTROLE is
      LECTEURS : INTEGER := 0;
 begin
      accept ECRIRE (X : in ITEM) do
                V:=X;
      end;
      loop
                select
                  accept DEBUT;
                  LECTEURS:=LECTEURS+1;
                or
                  accept FIN;
                  LECTEURS:=LECTEURS-1;
                or
                when LECTEURS = O =>
                accept ECRIRE (X : in ITEM) do
                   V := X;
                end;
               end select;
      end loop;
end CONTROLE;


procedure LIRE (X : out ITEM) is
begin
      CONTROLE.DEBUT;
      X :=V;
      CONTROLE.FIN;
end LIRE;
procedure ECRIRE (X : in ITEM) is
begin
      CONTRÔLE.ECRIRE(X);
end ECRIRE;
end LECTEURS_ECRIVAIN;
```

BTD/GL-ADA

15

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Donner la priorité à l'écrivain

- **Avec ECRIRE'COUNT = nombre de tâches couramment en attente dans la file de l'entrée ECRIRE**

```
select
        when ECRIRE'COUNT = 0 =>
        accept DEBUT;
        LECTEURS := LECTEURS + 1;
or
        accept FIN;
         LECTEURS := LECTEURS - 1;
or
        when LECTEURS = 0 =>
        accept ECRIRE (X: in ITEM) do
                V := X;
        end;
end select;
```

BTD/GL-ADA

16

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

SELECT avec DELAY

- **Si aucun appel à LIRE ou à ECRIRE n'est reçu dans un délai de 10 minutes, la troisième branche est prise et les instructions qui suivent l'instruction delay sont exécutées.**

```
select
        accept LIRE (…) do
        …
        end;
or
        accept ECRIRE (…) do
        …
        end;
or
        delay 10*MINUTES;
        -- instructions de dépassement d'attente
end select;
```

```
select
        OPERATEUR.APPEL (« ETEINDRE LE FEU »);
or
        delay 1*MINUTE;
        POMPIER (…);
end select;
```

BTD/GL-ADA

17

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

SELECT avec ELSE (1)

- **La dernière branche est précédée de ELSE et se compose simplement d'une suite d'instructions**

```
select
        accept LIRE (…) do
        …
        end;
or
        accept ECRIRE (…) do
        …
        end;
else
        -- instructions de remplacement
end select;
```

```
select
        OPERATEUR.APPEL (« ETEINDRE LE FEU »);
else
        POMPIER (…);
end select;
```

BTD/GL-ADA

18

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

SELECT avec ELSE (2)

- **Attention:** **Instruction SELECT ne peut avoir à la fois une partie ELSE et des alternatives de délai.**
- **Les appels d'entrée à attente limitée et conditionnels sont assez différents da l'instruction ACCEPT générale. Ils ne concernent que les appels isolés et sans garde :**
  - **les instructions SELECT ont toujours exactement deux branches - une avec l'appel d'entrée, l'autre avec la suite d'instructions de remplacement.**

BTD/GL-ADA

19

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Types tâches et activation

- **Il est quelquefois utile d'avoir plusieurs tâches semblables mais distinctes (éventuellement sans en prédire le nombre)**
- **Une déclaration de type tâche fournit un modèle pour des tâches semblables:**

```
task type T is
        entry E (...);
end ;
task body T is
        .....
end T;
```

- **Pour créer une vraie tâche, nous pouvons utiliser la forme normale des déclarations d'objets :**

```
X : T;
```

BTD/GL-ADA

20

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Déclarations

- **On peut aussi utiliser les objets tâches dans les types composés:**

```
TT: array (1..10) of T;
type ARTICLE is
        Record
                CT: T;
        end record;
```

- **On appelle les entrées de ces tâches en utilisant le nom de l'objet tâche :**

```
X.E (...);
TT(1).E(...);
A.CT.E(...);
```

- **Il est important de constater que les objets tâches ne sont pas des variables mais se comportent comme des constantes.**
- **Le type tâche est une forme de type limité (pas d'affectation et de comparaisons: =,?).**
- **On peut l'utiliser comme type effectif correspondant à un paramètre générique formel spécifié comme limité privé.**

BTD/GL-ADA 21

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Exécution

- **L'exécution d'une tâche est un processus en deux phases :**
  - **Activation : l'élaboration des déclarations du corps de tâche,**
  - **exécution proprement dite**
- **On peut aussi créer des tâches par l'intermédiaire de types accès :**

```
type REF_T is access T;
```

- **puis, nous pouvons créer une tâche en utilisant un allocateur :**

```
RX : REF_T := new T;
```

- **Le type REF_T est un type accès normal : l'assignation et les comparaisons d'égalité des objets sont autorisées.**

```
RX.E (...)
```

BTD/GL-ADA 22

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Terminaison et exceptions

- **Une tâche peut devenir achevée puis se terminer de diverses façons, y compris en atteignant sont END final.**
- **On est souvent en présence de boucles sans fin.**
- **Alternative TERMINATE dans SELECT : si le cadre dont dépend la tâche a atteint son END , est donc achevée, et si toutes les tâches soeurs et les tâches dépendantes sont terminées.**
- **L'ensemble entier se termine donc automatiquement.**
- **Instruction ABORT suivie d'une liste de noms de tâches : Arrêt violent !**

BTD/GL-ADA

23

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Exemple : Création d'agents

Contexte SERVEUR – UTILISATEURS et le principe de BOITE AUX LETTRES (en construction)

BTD/GL-ADA

24