14 - Rendez-vous

Programmation Concurrente - LI330
Université P. & M. Curie - année scolaire 2013/2014

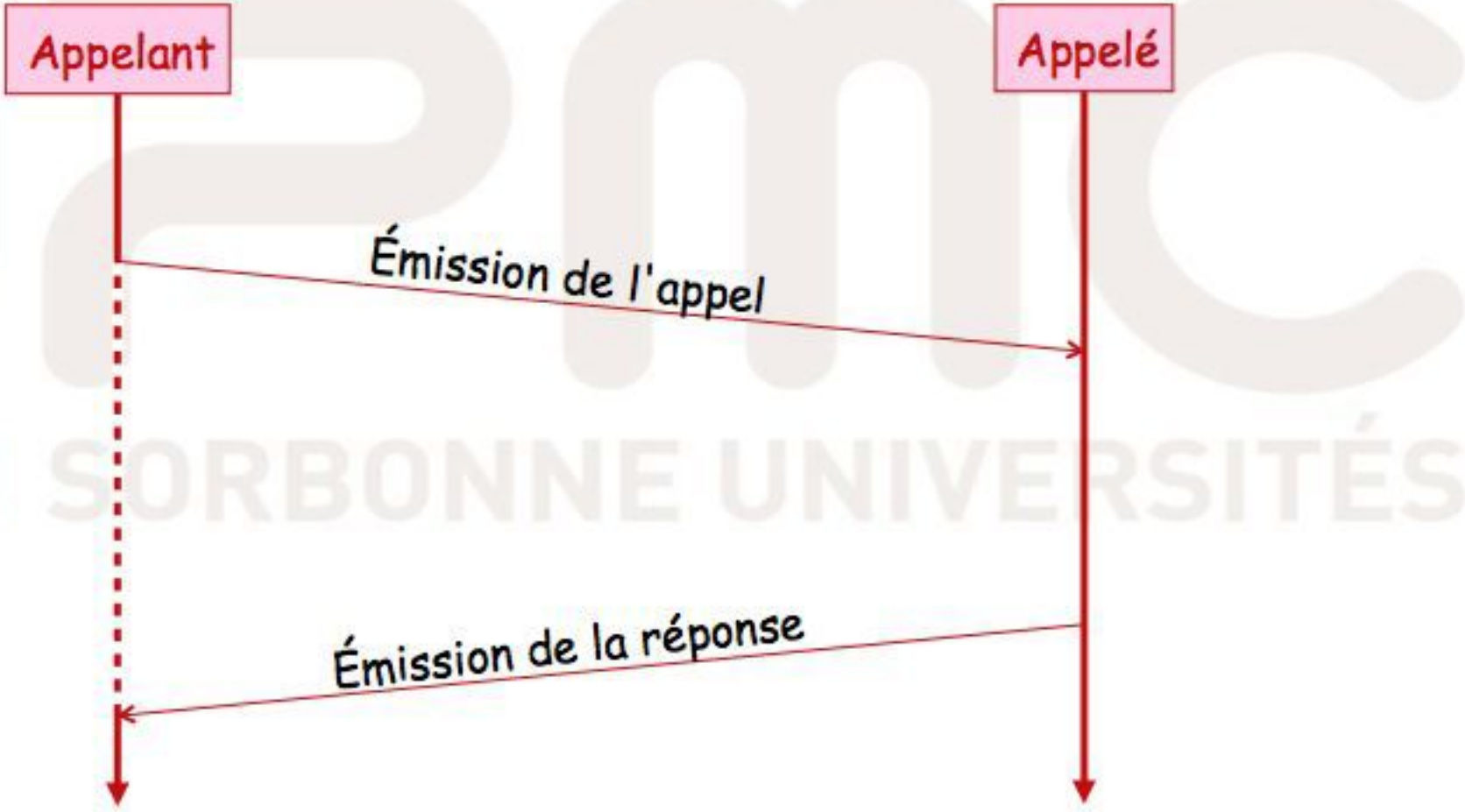
PrC

Le modèle du rendez-vous

Base: le modèle «RPC» (Remote Procedure Call)

RFC 707 (1976)

implémenté par Xerox et Sun MicroSystems au début des années 1980



Appel d'un point d'entré «comme une procédure»

```
<objet-tâche>.<point-d'entrée> (<paramètres>);
```

- Désignation de l'objet-tâche : nom, variable, pointeur, etc...

Attente d'un appel (blocage au sein d'un flux de contrôle)

```
accept <point-d'entrée> (<paramètres>) do
  <instructions>
end <point-d'entrée>;
```

- Visibilité des paramètres : do ... end
- Paramètres en mode in, in out et out

Exemple 1

```
with Ada.Text_Io;
use Ada.Text_Io;

procedure Com_Taches_1 is
   task Appelant;
   task Appele is
      entry Mon_Entree
            (X : in Natural);
   end Appele;

   task body Appelant is
   begin
      Appele.Mon_Entree (10);
   end Appelant;

   task body Appele is
   begin
      accept Mon_Entree
             (X : in Natural) do
         Put_Line ("Appelant a" &
                   Natural'Image (X));
      end Mon_Entree;
   end Appele;

begin
  null;
end Com_Taches_1;
```

File de processus

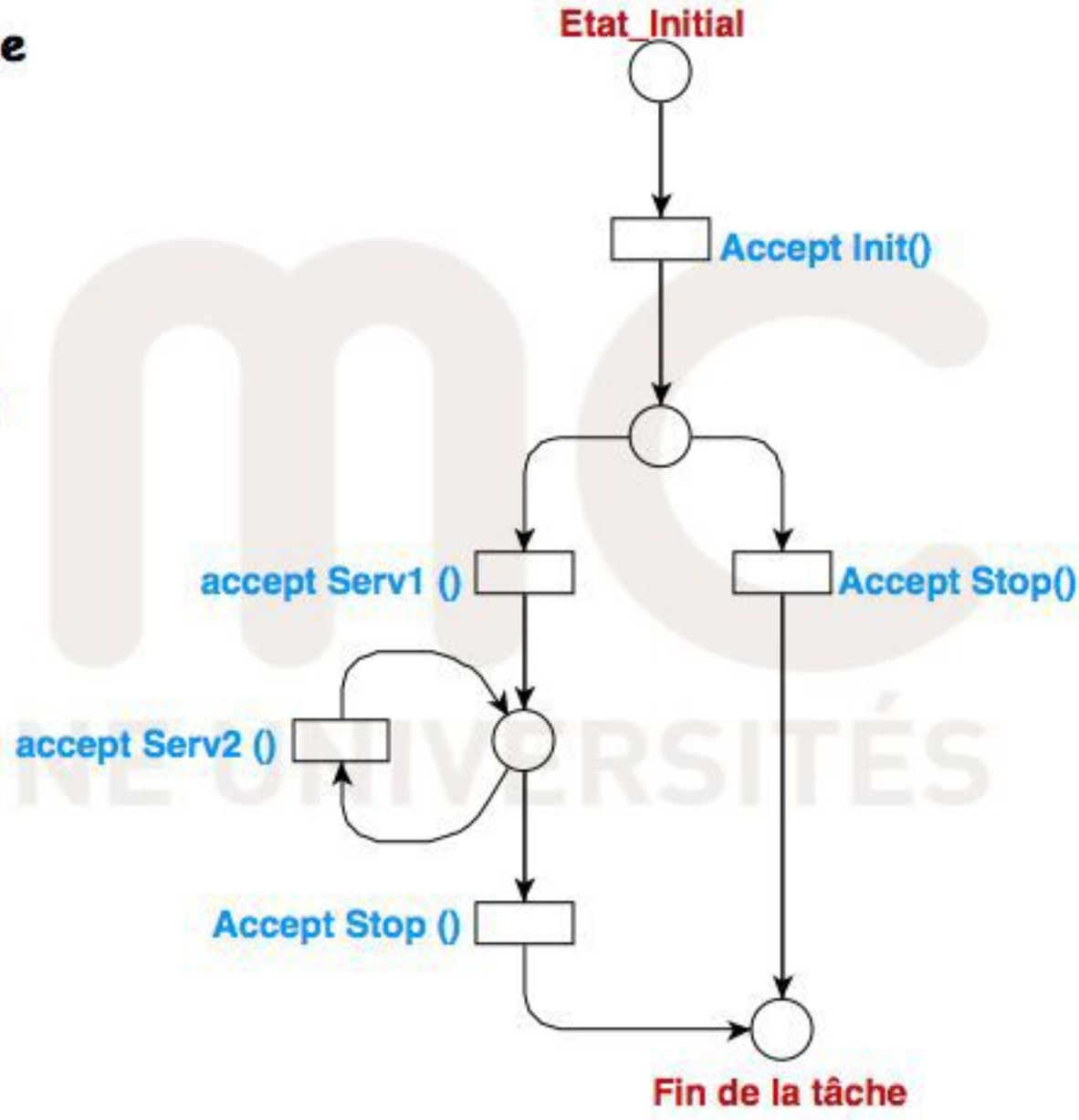
En attente de Mon_Entree

Appel indivisible

Provoque la suspension de l'appelant

Flux d'exécution et machines à états

- Un flux d'exécution est une «machine à états»
  - Un seul compteur ordinal
- Blocage sur les requêtes provenant d'un point d'entrée => des besoins
  - Possibilité d'alternatives
  - Possibilité de délais de temporisation
  - Possibilité de lecture conditionnelle
- Exemple d'automate
  - Indique la gestion des attentes bloquantes
  - Outils pour comprendre la dynamique d'un flux d'exécution
- Besoin d'un mécanisme dédié
  - Instruction select
    - Attente multiple
    - Attente temporisée
    - Attente conditionnelle



L'instruction select

- Objectif, gérer des attentes multiples et complexes
- Syntaxe:

```
un_select       select
                   alt_de_select
                {or
                   alt_de_select}
                [else
                   suite_d_instructions]
                end select;
alt_select      [when condition =>] une_attente
une_attente     altern_accept | altern_delay | altern_term
altern_accept   instruction_accept [suite_d_instructions]
altern_delay    delay valeur_flotante [suite_d_instructions]
altern_term     terminate
```

Exemple 1 (1)

```
with Ada.Text_Io;
use Ada.Text_Io;

procedure Ex1_Accept is

   task type Client is
      entry Initialise (Id : in Natural);
   end Client;

   task Serveur is
      entry Initialise (Val : in Natural);
      entry Adios;
      entry Service (Qui : in Natural);
   end Serveur;
```

../..

Exemple 1 (2)

```
task body Client is
   My_Id : Natural;
begin
   -- Initialisation
   accept Initialise (Id : in Natural) do
      My_Id := Id;
   end Initialise;
   -- Appel au serveur
   loop
      Serveur.Service (My_Id);
   end loop;
 -- Ne se termine pas...
end Client;
```


../..

Exemple 1 (3)

```
task body Serveur is
   Max_Service : Integer;
   Rester : Boolean := True;
begin
   -- Initialisation
   accept Initialise (Val : in Natural) do
      Max_Service := Val;
   end Initialise;
```


../..

Exemple 1 (4)

```
-- je rend le service...
while Rester loop
   select
      accept Service (Qui : in Natural) do
         Put_Line ("Service pour" & Natural'Image (Qui));
           Max_Service := Max Service -1;
      end Service;
   or
      when Max_Service <= 0 =>
         accept Adios;
           Put_Line ("Je me termine");
           Rester := False;
   end select;
end loop;
end Serveur;
```

Boucle de service (archi-classique;-)

Indéterminisme entre les alternatives de select

Peut-être déclenché si Max_Service ≤ 0

../..

Exemple 1 (5)

```
   Les_Clients : array (1..5) of Client;

Begin -- Ex1_Accept
   -- Lancer les clients avec une identite
   for I in Les_Clients'Range loop
      Les_Clients(I).Initialise (I);
   end loop;
   -- lancer le serveur
   Serveur.Initialise (7);
   -- Attendre qu'il se termine
   Serveur.Adios;
end Ex1_Accept;
```

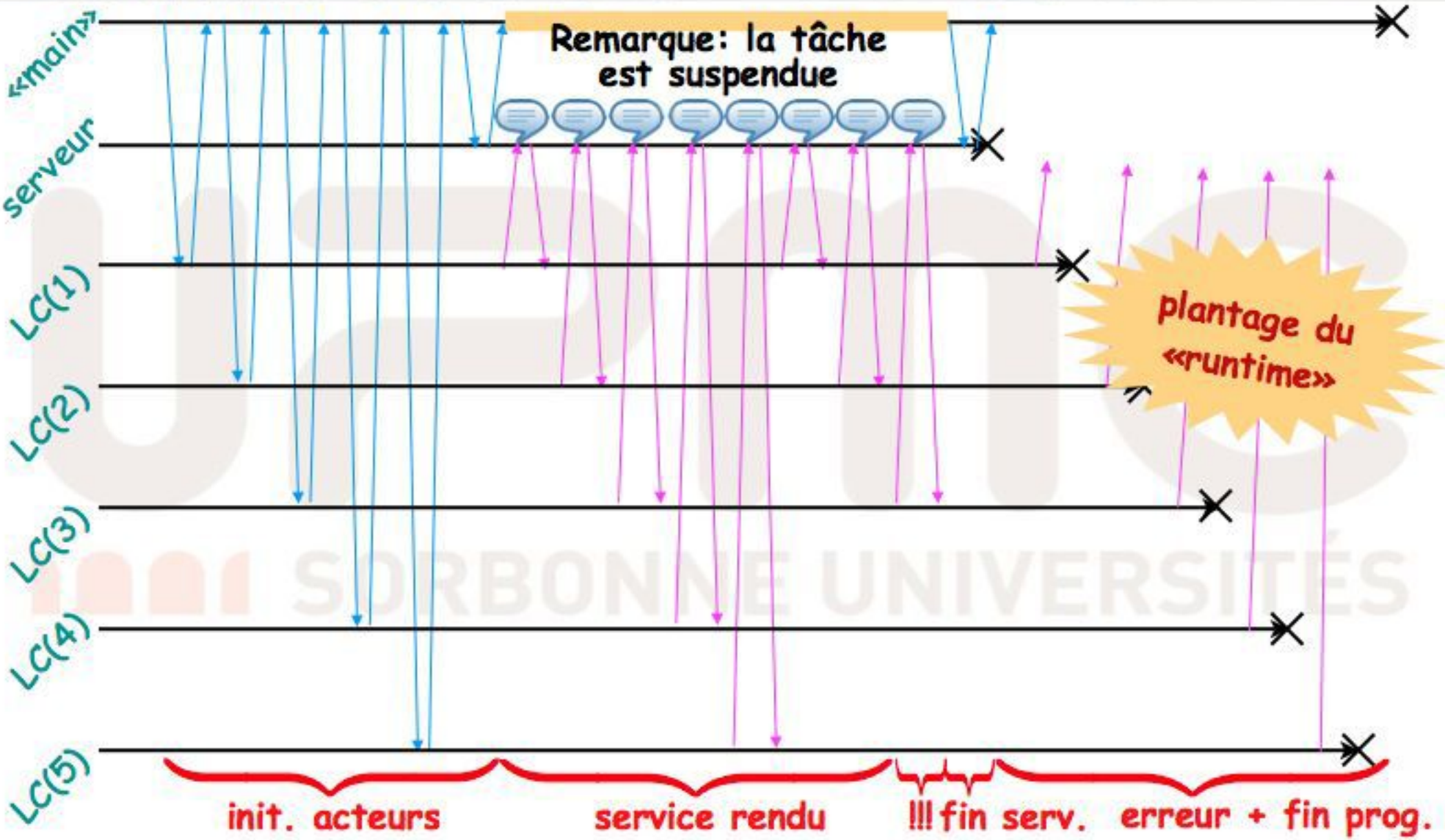
Commutations...

```
$ ./ex1_accept
Service pour 1
Service pour 2
Service pour 3
Service pour 4
Service pour 5
Service pour 1
Service pour 2
Service pour 3
raised TASKING_ERROR
```

Rappel, le «main» est une tâche aussi

Levée dès que Serveur se termine

Analyse de l'exécution au moyen d'un chronogramme



Exemple 2 (1)

- Nouveau client, serveur et programme principal
  - Assurer proprement la terminaison du programme

```
task body Client is
    My_Id : Natural;
begin
    -- Initialisation
    accept Initialise (Id : in Natural) do
       My_Id := Id;
    end Initialise;
    -- Appel au serveur
    for I in 1 .. 2 loop
       Serveur.Service (My_Id);
    end loop;
end Client;
```

../..

Exemple 2 (2)

```
task Serveur is
   entry Initialise (Val : in Natural);
   entry Service (Qui : in Natural);
end Serveur;

task body Serveur is
   Max_Service : natural;
begin
   -- Initialisation
   accept Initialise (Val : in Natural) do
      Max_Service := Val;
   end Initialise;
   -- je rend le service...
   loop
      select
         accept Service (Qui : in Natural) do
            Put_Line ("Service pour" & Natural'Image (Qui));
            Max_Service := Max_Service -1;
         end Service;
      or
         terminate;
      end select;
   end loop;
end Serveur;
```

../..

Exemple 2 (3)


```
   Les_Clients : array (1..5) of Client;

begin -- Ex3_Accept
   -- Lancer les clients avec une identite
   for I in Les_Clients'Range loop
      Les_Clients(I).Initialise (I);
   end loop;
   -- lancer le serveur (trop d'apels nécessaire pour
   -- qu'il se termine)
   Serveur.Initialise (30); -- discriminant inutile!
   -- Attendre qu'il se termine
end Ex3_Accept;
```

```
./ex3_accept
Service pour 1
Service pour 2
Service pour 3
Service pour 4
Service pour 5
Service pour 1
Service pour 2
Service pour 3
Service pour 4
Service pour 5
```

Exemple 3 (1)

Illustration des temporisations

```
task body Client is
   My_Id : Natural;
begin
   -- Initialisation
   accept Initialise (Id : in Natural) do
      My_Id := Id;
   end Initialise;
   -- Appel au serveur
   for I in 1 .. 5 loop
      Serveur.Service (My_Id);
      delay 3.0 * Duration(My_Id);
   end loop;
end Client;
```

../..

Exemple 3 (2)

```
task body Serveur is
     Max_Service : natural;
  begin
     -- Initialisation
     accept Initialise (Val : in Natural) do
       Max_Service := Val;
     end Initialise;
     -- je rend le service...
     while Max_Service > 0 loop
        select
           accept Service (Qui : in Natural) do
              Put_Line ("Service pour" & Natural'Image (Qui));
              Max_Service := Max_Service - 1;
           end Service;
        or
          delay 2.0;
           Put_Line ("Soeur Anne, ne vois-tu rien venir?");
        end select;
   end loop;
  end Serveur;
```

../..

Exemple 3 (3)


```
Les_Clients : array (1..2) of Client;

begin – Ex4_Accept
   -- Lancer les clients avec une identite
   for I in Les_Clients'Range loop
     Les_Clients(I).Initialise (I);
   end loop;
   -- lancer le serveur
   Serveur.Initialise (10);
   -- Attendre qu'il se termine
end Ex4_Accept;
```

```
$ ./ex4_accept
Service pour 1
Service pour 2
Soeur Anne, ne vois-tu rien venir?
Service pour 1
Soeur Anne, ne vois-tu rien venir?
Service pour 2
Service pour 1
Soeur Anne, ne vois-tu rien venir?
Service pour 1
Soeur Anne, ne vois-tu rien venir?
Service pour 2
Service pour 1
Soeur Anne, ne vois-tu rien venir?
Soeur Anne, ne vois-tu rien venir?
Service pour 2
Soeur Anne, ne vois-tu rien venir?
Soeur Anne, ne vois-tu rien venir?
Service pour 2
```

Exemple 4 (1)

Modification du client et du «main»

- Montrer l'usage de l'alternative else

```
task body Client is
      My_Id : Natural;
   begin
      -- Initialisation
      accept Initialise (Id : in Natural) do
         My_Id := Id;
      end Initialise;
      -- Appel au serveur
      for I in 1 .. 5 loop
         select
            Serveur.Service (My_Id);
         else
            Put_Line ("Il se moque de moi!");
         end select;
      end loop;
   end Client;
```

Select marche aussi pour l'appelant (**sauf «terminate»**), **pour une entrée** avec un **«or delay»** ou un **«else»**)

../..

Exemple 4 (2)

```
Les_Clients : array (1..2) of Client;

begin -- Ex5_Accept
   -- lancer le serveur
   Serveur.Initialise (10);
   -- Lancer les clients avec une identite
   for I in 1..2 loop -- Les_Clients'Range
      Les_Clients(I).Initialise (I);
   end loop;
end Ex5_Accept;
```

```
$ ./ex5_accept
Il se moque de moi!
Il se moque de moi!
Il se moque de moi!
Il se moque de moi!
Il se moque de moi!
Service pour 1
Service pour 1
Service pour 1
Service pour 1
Service pour 1
^C
```

Déclenchement de l'alternative else dans le select

Select dans des appels de points d'entrée: remarques (j'insiste ;-)

- Toutes les combinaisons ne sont pas possibles
  - La directive terminate n'a pas de sens
  - Les gardes non plus
- Cas 1: time-out

```
select
        Serveur.Service (Mes_Parametres);
or delay t;
end select;
```

- Cas 2: absence de l'interlocuteur (time-out avec t=0)

```
select
        Serveur.Service (Mes_Parametres);
else
        Faire_Quelque_Chose;
end select;
```

Attributs prédéfinis du langage

- Sur les tâches
  - <tâche>'Callable
    - Valeur booléenne indiquant si une tâche peut être contactée (i.e. n'est pas terminée ni «plantée»)
  - <tâche>'Terminated
    - Valeur booléenne indiquant si une tâche est terminée
- Sur les points d'entrée de tâche
  - <entrée>'Count
    - Rend le nombre de requêtes dans la file associée au point d'entrée
    - Possible seulement dans le corps d'une tâche