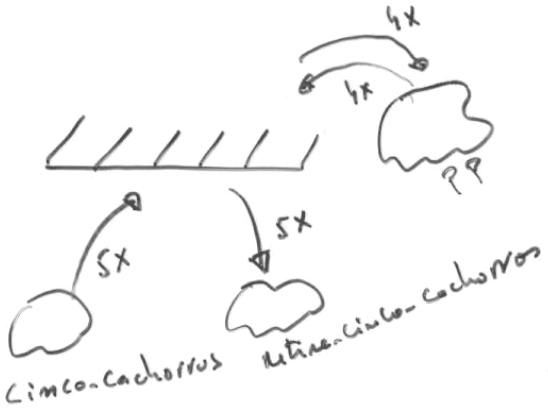
```
with Ada.Text_Io, Ada.Integer_Text_Io;
use Ada.Text_Io, Ada.Integer_Text_Io;

procedure Armazem is

   Numero_Cachorros : Integer := 0;

   -- neste caso existe um local onde são colocados os cachorros quentes
   -- e de onde são retirados quando um cliente pede cachorros quentes

   task Cinco_Cachorros;         -- esta tarefa adiciona cinco cachorros ao stock
   task Retira_Cinco_Cachorros; -- esta tarefa retira cinco cachorros ao stock

   task Armazenamento_Cachorros is
      entry Aumentar_Stock;         -- adiciona um cachorro ao stock
      entry Diminuir_stock;        -- apaga um cachorro do stock
   end Armazenamento_Cachorros;

   task body Armazenamento_Cachorros is
   begin
      for Index in 1..18 loop
         Put("estamos no loop ");
         put(index);
         select
            when Numero_Cachorros < 5 =>
            accept Aumentar_Stock do
               Numero_Cachorros:= Numero_Cachorros + 1;
               Put("adicionamos um cachorro ao stock. Temos agora ");
               Put(Numero_Cachorros, 3);
               New_Line;
            end Aumentar_Stock;
         or
            when Numero_Cachorros > 0 =>
            accept Diminuir_stock do
               Put_Line("retirar um cachorro do stock");
               Numero_Cachorros := Numero_Cachorros - 1;
            end Diminuir_stock;
         end select;
      end loop;
   end Armazenamento_Cachorros;

   task body Cinco_Cachorros is
   begin
      for Index in 1..5 loop
         delay 0.1;
         Armazenamento_Cachorros.Aumentar_Stock;
      end loop;
   end Cinco_Cachorros;

   task body Retira_Cinco_Cachorros is
   begin
      for Index in 1..5 loop
         delay 0.6;
         Armazenamento_Cachorros.Diminuir_Stock;
      end loop;
   end Retira_Cinco_Cachorros;

begin
   for Index in 1..4 loop
      delay 0.9;
      Armazenamento_Cachorros.Aumentar_Stock;
      Armazenamento_Cachorros.Diminuir_Stock;
   end loop;
end armazem;
```



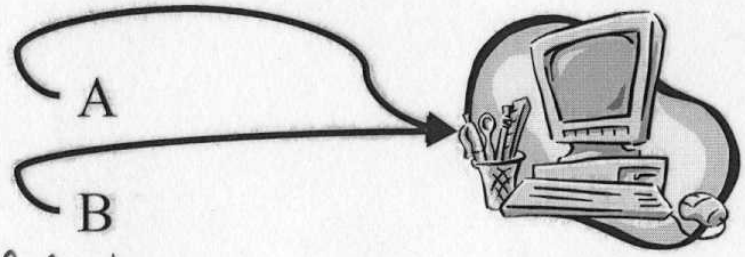
Select
when
accept
or
⋮
end select

Rendezvous para exclusão mútua

```
task SCREEN_ACCESS is
  entry PUT (S : in STRING);
end SCREEN_ACCESS;

task body SCREEN_ACCESS is
begin
  loop
    accept PUT (S : in STRING) do
      ADA.TEXT_IO.PUT (S);
    end PUT;
  end loop;
end SCREEN_ACCESS;
```

```
Task A;

Task body A is

Begin

loop

Faz_qq_coisa;

...

Screen_access.put ('ola');

Faz_qq_coisa;

...

End loop;

End;
```

```
Task B;

Task body B is

Begin

loop

Faz_qq_coisa;

...

Screen_access.put ('Bom dia');

Faz_qq_coisa;

...

End loop;

End;
```



Programa c/ estes 3 entidades!

Termina?

Rendezvous para sincronização

```
task SCREEN_ACCESS is
  entry ENTER;
  entry LEAVE;
end SCREEN_ACCESS;

task body SCREEN_ACCESS is
begin
  loop
    accept ENTER;
    accept LEAVE;
  end loop;
end SCREEN_ACCESS;
```

```
Task A;
Task body A is
Begin
loop
Faz_qq_coisa;
...
Screen_access.enter;
Faz_qq_coisa;
...
End loop;
End;
```

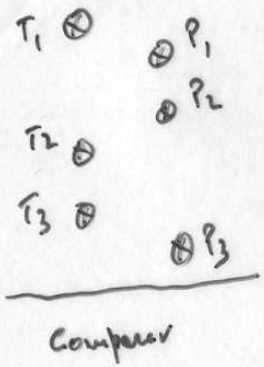
Ler 3 Temp. Votar

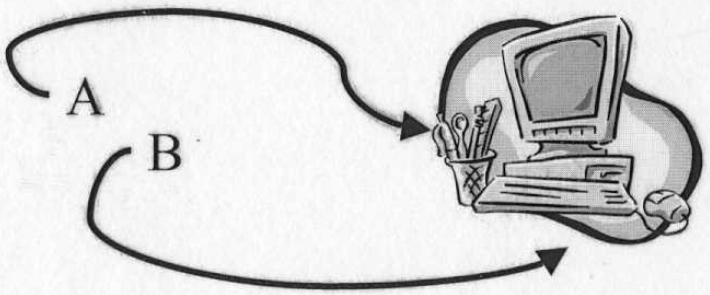
Comparar T/P

```
Task A;
Task body A is
Begin
loop
Faz_qq_coisa;
...
Screen_access.leave;
Faz_qq_coisa;
...
End loop;
End;
```

Ler 3 Pressões Votar

Comparar P/T



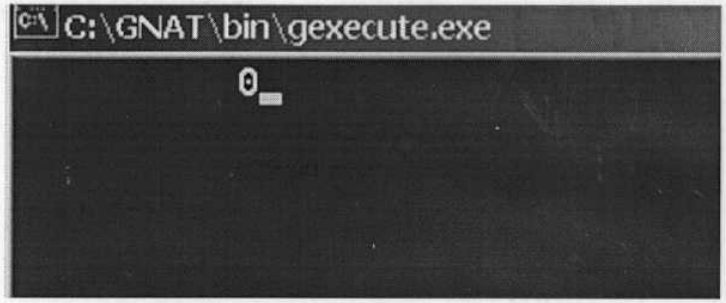


Um caso conhecido...

```
task ACCESS_CONTROL is
  entry ADD (N : in INTEGER);
  entry SUBTRACT (N : in INTEGER);
  entry VALUE (N : out INTEGER);
end ACCESS_CONTROL;

task body ACCESS_CONTROL is

  VALUE : INTEGER := 0;

begin
  loop
    accept ADD (N : in INTEGER) do
      VALUE := VALUE + N;
    end ADD;
    accept SUBTRACT (N : in INTEGER) do
      VALUE := VALUE - N;
    end SUBTRACT;
    accept VALUE (N : out INTEGER) do
      N := VALUE;
    end VALUE;
  end loop;
end ACCESS_CONTROL;
```

```
Task add;

Task body add is
Begin
Access_control.ADD(1);
End;

Task subtract;

Task body subtract is
Begin
Access_control.subtract (1);
End;

Task value;

Task body value is
Begin
Access_control.value (N);
End;
```

```
with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;

procedure Dado_Partilhado is

   N: Integer;
   task Access_Control is
      entry Add (Value:in Integer);
      entry Subtract (Value:in Integer);
      entry Value(Value:out Integer);
   end Access_Control;

   task body Access_Control is
      N:Integer:=0;
   begin
      loop
         accept Add (Value: in Integer) do
            N:=N+Value;
         end Add;
         accept Subtract (Value: in Integer) do
            N:=N-Value;
         end Subtract;
         accept Value (Value: out Integer) do
            Value:=N;
         end Value;
      end loop;
   end;

   task Decrement;
   task body Decrement is
   begin
      Access_Control.Subtract(1);
   end Decrement;

   task Increment;
   task body Increment is
   begin
      Access_Control.Add(1);
   end Increment;

begin
   Access_Control.Value(N);
   Put(N);
end Dado_Partilhado;
```



Uma alternativa... Mas cuidado com os resultados...

```
task ACCESS_CONTROL is
  entry ADD (N : in INTEGER);
  entry SUBTRACT (N : in INTEGER);
  entry VALUE (N : out INTEGER);
end ACCESS_CONTROL;

task body ACCESS_CONTROL is

  VALUE : INTEGER := 0;

begin
  loop
    select
      accept ADD (N : in INTEGER) do
        VALUE := VALUE + N;
      end ADD;
    or
      accept SUBTRACT (N : in INTEGER) do
        VALUE := VALUE - N;
      end SUBTRACT;
    or
      accept VALUE (N : out INTEGER) do
        N := VALUE;
      end VALUE;
    end select;
  end loop;
end ACCESS_CONTROL;
```

A operação select só pode seleccionar entre tratamentos select

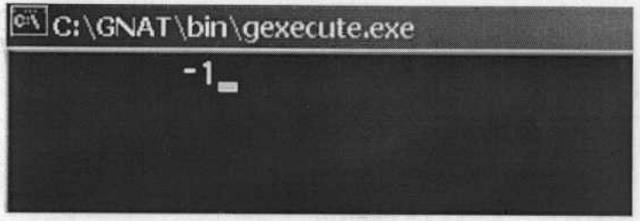
Apesar de um accept ter de ser o primeiro, outras declarações podem seguir-se

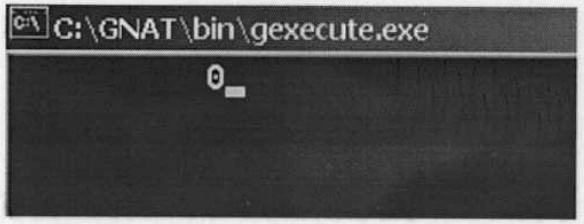
A clausula or separa as várias alternativas. Mais alternativas que as apresentadas são permitidas

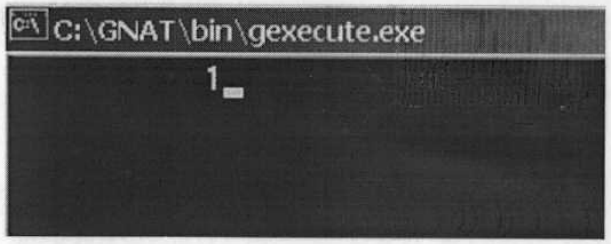
```
Task add;

Task body add is
Begin
Access_control.ADD(1);
End;

Task subtract;

Task body subtract is
Begin
Access_control.subtract (1);
End;

Task value;

Task body value is
Begin
Access_control.value (N);
End;
```

Vantagens? Desvantagens? Perigos?

```
with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;

procedure Dado_Partilhado is

   N: Integer;
   task Access_Control is
      entry Add (Value:in Integer);
      entry Subtract (Value:in Integer);
      entry Value(Value:out Integer);
   end Access_Control;

   task body Access_Control is
      N:Integer:=0;
   begin
      loop
         select
            accept Add (Value: in Integer) do
               N:=N+Value;
            end Add;
         or
            accept Subtract (Value: in Integer) do
               N:=N-Value;
            end Subtract;
         or
            accept Value (Value: out Integer) do
               Value:=N;
            end Value;
         end select;
      end loop;
   end;

   task Decrement;
   task body Decrement is
   begin

      Access_Control.Subtract(1);
   end Decrement;

   task Increment;
   task body Increment is
   begin
      Access_Control.Add(1);
   end Increment;

begin

   Access_Control.Value(N);
   Put(N);
end Dado_Partilhado;
```







Solução???

```
with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;

procedure Dado_Partilhado is

   N: Integer;
   task Access_Control is
      entry Add (Value:in Integer);
      entry Subtract (Value:in Integer);
      entry Value(Value:out Integer);
   end Access_Control;

   task body Access_Control is
      N:Integer:=0;
   begin
      loop
         select
            accept Add (Value: in Integer) do
               N:=N+Value;
            end Add;

         or
            accept Subtract (Value: in Integer) do
               N:=N-Value;
            end Subtract;
         or
            accept Value (Value: out Integer) do
               Value:=N;
            end Value;
         end select;
      end loop;
   end;

   task Decrement;
   task body Decrement is
   begin
      Access_Control.Subtract(1);
   end Decrement;

   task Increment;
   task body Increment is
   begin
      Access_Control.Add(1);
   end Increment;

begin

   if Decrement'Terminated and Decrement'Terminated then
      Access_Control.Value(N);
      Put(N);
   end if;
end Dado_Partilhado;
```

loop

exit

End loop

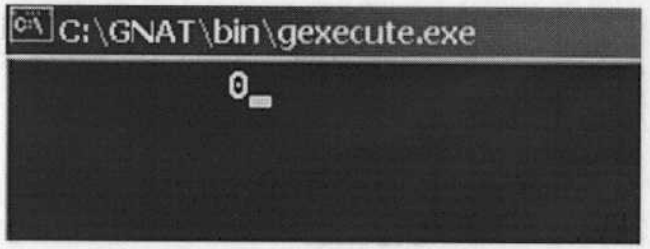
```
with Ada.Integer_Text_Io, Ada.Text_Io;
use Ada.Integer_Text_Io, Ada.Text_Io;

procedure Dado_Partilhado is

   N: Integer:=0;
   task Access_Control is
      entry Add (Value:in Integer);
      entry Subtract (Value:in Integer);
      entry Value(Value:out Integer);
   end Access_Control;

   task body Access_Control is
      N:Integer:=0;
   begin
      loop
         select
            accept Add (Value: in Integer) do
               N:=N+Value;
            end Add;
         or
            accept Subtract (Value: in Integer) do
               N:=N-Value;
            end Subtract;
         or
            accept Value (Value: out Integer) do
               Value:=N;
            end Value;
         end select;
      end loop;
   end;

   task Decrement;
   task body Decrement is
   begin
      Access_Control.Subtract(1);
   end Decrement;

   task Increment;
   task body Increment is
   begin
      Access_Control.Add(1);
   end Increment;

   task Valor;
   task body Valor is
   begin
      while not (Decrement'Terminated and Increment'Terminated) loop
      null;
      end loop;
      Access_Control.Value(N);
      Put(N);
        end;

begin
   null;
   end Dado_Partilhado;
```

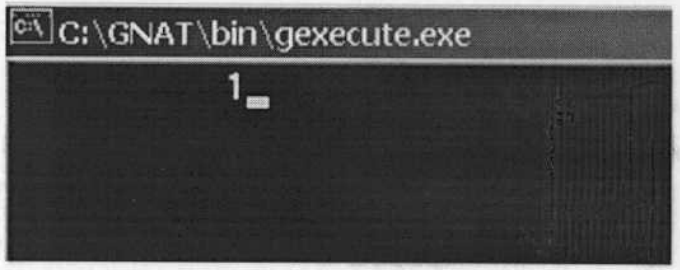
As alternativas com guardas

```
TASK ACCESS_CONTROL is
 entry ADD(N: in INTEGER);
 entry SUBTRACT (N: in INTEGER);
 entry VALUE  (N: OUT INTEGER);
end ACCESS_CONTROL;

 Task body Access_control is
  VALUE : INTEGER := 0
begin
 loop
  Select
   accept  ADD (N: in INTEGER) d
     VALUE := VALUE + N;
   end ADD;
 or
 when N > 0 =>
 accept Substract
```

```
Task add;

Task body add is
Begin
Access_control.ADD(1);
End;

Task subtract;

Task body subtract is
Begin
Access_control.subtract (1);
End;

Task value;

Task body value is
Begin
Access_control.value (N);
End;
```

```
with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;

procedure Dado_Partilhado is

   N: Integer;
   task Access_Control is
      entry Add (Value:in Integer);
      entry Subtract (Value:in Integer);
      entry Value(Value:out Integer);
   end Access_Control;

   task body Access_Control is
      N:Integer:=0;
   begin
      loop
         select
            accept Add (Value: in Integer) do
               N:=N+Value;
            end Add;

         or
            when N>0 =>
            accept Subtract (Value: in Integer) do
               N:=N-Value;
            end Subtract;
         or
            accept Value (Value: out Integer) do
               Value:=N;
            end Value;
         end select;
      end loop;
   end;

   task Decrement;
   task body Decrement is
   begin
      Access_Control.Subtract(1);
   end Decrement;

   task Increment;
   task body Increment is
   begin
      Access_Control.Add(1);
   end Increment;

begin
   Access_Control.Value(N);
   Put(N);

end Dado_Partilhado;
```





0(-1)? Já não sei!

PG?

Outras alternativas de um select

O ELSE

Uma instrução select pode ter uma parte final na forma:

```
select
   alternativa
or
   alternativa
else
sequência de instruções
end select
```

A parte else executa-se se ao chegar-se ao select não se puder aceitar de imediato mais nenhuma alternativa.

Não podemos ter else e alternativas temporizadas ao mesmo tempo dentro de um select.

A parte else não é uma alternativa e portanto não pode ser guardada.

```
with Ada.Integer_Text_Io, Ada.Text_Io;
use Ada.Integer_Text_Io, Ada.Text_Io;

procedure Dado_Partilhado is

   N: Integer;
   task Access_Control is
      entry Add (Value:in Integer);
      entry Subtract (Value:in Integer);
      entry Value(Value:out Integer);
   end Access_Control;

   task body Access_Control is
      N:Integer:=0;
   begin
      loop
         delay 2.0;
         select
            accept Add (Value: in Integer) do
               N:=N+Value;
               Put_Line("accept add");
            end Add;
         or
            when N>0 =>
            accept Subtract (Value: in Integer) do
               N:=N-Value;
               Put_Line("accept subtract");
            end Subtract;
         or
            accept Value (Value: out Integer) do
               Value:=N;
               Put_Line("accept subtract");
            end Value;
        else
           Put_line ("nenhum pedido");
         end select;
      end loop;
   end;
```

```
   task Decrement;
   task body Decrement is
   begin
      delay 2.5;
      Access_Control.Subtract(1);
   end Decrement;

   task Increment;
   task body Increment is
   begin
      delay 2.5;
      Access_Control.Add(1);
   end Increment;

begin
   delay 8.0;
   Access_Control.Value(N);
   Put(N);
end Dado_Partilhado;
```

A ALTERNATIVA TERMINATE

Uma das alternativas de um select pode ter a forma :

terminate;

Esta alternativa é seleccionada quando:

- Todas as tarefas que fazem chamadas para as entradas da tarefa servidora (Masters) tiverem terminado.

- Cada tarefa que depende do Master considerado tenha também já terminado, ou aguarde bloqueado num tratamento select com uma alternativa terminate aberta.

- É conveniente que as tarefas servidoras terminem assim.

- A alternativa terminate pode estar guardada por guardas

- É incompatível com as alternativas temporizadas e com a parte else

```
task MAILBOX is
  entry POST (I : in INTEGER);
  entry COLLECT (I : out INTEGER);
end MAILBOX;

task body MAILBOX is
  BOX : INTEGER;
  FREE : BOOLEAN := TRUE;
begin
  loop
    select
      when FREE =>
        accept POST (I : in INTEGER) do
          BOX := I;
        end POST;
        FREE := FALSE;
    or
      when not FREE =>
        accept COLLECT (I : out INTEGER) do
          I := BOX;
        end POST;
        FREE := TRUE;
    or
      terminate;
    end select;
  end loop;
end MAILBOX;
```

```
with Ada.Integer_Text_Io, Ada.Text_Io;
use Ada.Integer_Text_Io, Ada.Text_Io;

procedure Dado_Partilhado is

   N: Integer;
   task Access_Control is
      entry Add (Value:in Integer);
      entry Subtract (Value:in Integer);
      entry Value(Value:out Integer);
   end Access_Control;

   task body Access_Control is
      N:Integer:=0;
   begin
      loop
         delay 2.0;
         select
            accept Add (Value: in Integer) do
               N:=N+Value;
               Put_Line("accept add");
            end Add;
         or
            when N>0 =>
            accept Subtract (Value: in Integer) do
               N:=N-Value;
               Put_Line("accept subtract");

            end Value;
         or
            terminate;
         end select;
      end loop;
   end;

   task Decrement;
   task body Decrement is
   begin
      delay 2.5;
      Access_Control.Subtract(1);
   end Decrement;

   task Increment;
   task body Increment is
   begin
      delay 2.5;
      Access_Control.Add(1);
   end Increment;

begin
   delay 8.0;
   Access_Control.Value(N);
   Put(N);
end Dado_Partilhado;
```

A alternativa delay

```
...
loop
    select
        when FREE =>
            accept POST (I:in Integer) do
                box:=I;
            end POST;
            FREE := FALSE;
        or
        when not FREE =>
            accept COLLECT (I:out Integer) do
                I:= box;
            end COLLECT;
            FREE := TRUE;
        or
        when not FREE =>
        delay 5.0;
        FREE := TRUE;
        or
        delay 10.0;
        Put_Line ("No data present");
    end select
end loop;
...
```

Os timeouts podem ter guardas

Quando dados presentes em box não são recolhido num prazo de 5 segundos, limpa-se a variável.

Se nenhum dado for colocado na variável box durante 10 segundos, estando ela livre, emite-se uma mensagem de erro

```
with Ada.Integer_Text_Io, Ada.Text_Io;
use Ada.Integer_Text_Io, Ada.Text_Io;

procedure Dado_Partilhado is

   N: Integer;
   task Access_Control is
      entry Add (Value:in Integer);
      entry Subtract (Value:in Integer);
      entry Value(Value:out Integer);
   end Access_Control;

   task body Access_Control is
      N:Integer:=0;
   begin
      loop
         select
            accept Add (Value: in Integer) do
               N:=N+Value;
               Put_Line("accept add");
            end Add;
         or
            when N>0 =>
            accept Subtract (Value: in Integer) do
               N:=N-Value;
               Put_Line("accept subtract");
            end Subtract;
         or
            accept Value (Value: out Integer) do
               Value:=N;
               Put_Line("accept subtract");
            end Value;
         or
            delay 5.0;
            Put_Line ("nao chegou qualquer pedido");
         end select;
      end loop;
   end;

   task Decrement;
   task body Decrement is
   begin
      delay 2.5;
      Access_Control.Subtract(1);
   end Decrement;

   task Increment;
   task body Increment is
   begin
      delay 2.5;
      Access_Control.Add(1);
   end Increment;

begin
   delay 8.0;
   Access_Control.Value(N);
   Put(N);
end Dado_Partilhado;
```