Problema Microkenels:
Filósofos y tenedores, solución con jerarquías en Ada
por Lizandro Ramirez.

En este documento se presenta la solución del famoso problema informático sobre los filósofos que se sientan a comer y necesitan compartir tenedores, puede encontrar una explicación detallada del problema en (http://en.wikipedia.org/wiki/Dining_philosophers_problem). La solución planteada es la conocida como "Resource hierarchy solution" esta solución al problema es el propuesto originalmente por Dijkstra. La cual Asigna un orden parcial de los recursos (los tenedores, en este caso) . En este caso, se numerarán los recursos (tenedores) serán del 1 al 5 y cada unidad de trabajo (filósofo) siempre coger el tenedor número más bajo primero, y luego tenedor numeración más alta entre los tenedores que están a su alcance en la mesa y planea utilizar . El orden en que cada filósofo coloca las los tenedores no importa. En este caso, si cuatro de los cinco filósofos recogen simultáneamente su tenedor numerado menor, sólo el tenedor con el número más alto permanecerá sobre la mesa, por lo que el quinto filósofo no será capaz de recoger ningún tenedor . Por otra parte , sólo un filósofo tendrá acceso a dicho tenedor de número más alto , por lo que será capaz de comer usando dos tenedores.

Ejemplo de la prueba realizada en Ubuntu se puede ver que todos llegan a comer:

```
lizandro@lizandro-laptop:~/Dropbox/Sistemas Empotrados/MicroKenels/Los_filosofos_ada$ ./pru_filosofo_tenedor
filosofo1 toma el tenedor1 y medito
filosofo5 toma el tenedor4 y medito
filosofo3 toma el tenedor2 y medito
filosofo4 toma el tenedor3 y medito
filosofo1 toma el tenedor5 y come
filosofo1 suelto tenedor 1 y 5
filosofo2 toma el tenedor1 y medito
filosofo5 toma el tenedor5 y come
filosofo5 suelto tenedor 4 y 5
filosofo4 toma el tenedor4 y come
filosofo4 suelto tenedor 3 y 4
filosofo3 toma el tenedor3 y come
filosofo3 suelto tenedor 2 y 3
filosofo2 toma el tenedor2 y come
filosofo4 toma el tenedor3 y medito
filosofo5 toma el tenedor4 y medito
filosofo2 suelto tenedor 1 y 2
filosofo1 toma el tenedor1 y medito
filosofo3 toma el tenedor2 y medito
filosofo1 toma el tenedor5 y come
filosofo1 suelto tenedor 1 y 5
filosofo2 toma el tenedor1 y medito
filosofo5 toma el tenedor5 y come
filosofo5 suelto tenedor 4 y 5
filosofo4 toma el tenedor4 y come
filosofo4 suelto tenedor 3 y 4
filosofo3 toma el tenedor3 y come
filosofo5 toma el tenedor4 y medito
filosofo3 suelto tenedor 2 y 3
filosofo2 toma el tenedor2 y come
filosofo4 toma el tenedor3 y medito
filosofo2 suelto tenedor 1 y 2
filosofo1 toma el tenedor1 y medito
filosofo3 toma el tenedor2 y medito
```

1- Codigo de pru_filosofo_tenedor.adb usando para inicial la el programa:

```
with Filosofo_Tenedor;
use Filosofo_Tenedor;

procedure Pru_Filosofo_Tenedor is
   pragma Priority(0);
begin
   Ocioso;
end Pru_Filosofo_Tenedor;
```

2- Codigo de filosofo_tenedor.ads:

```
-- Prueba de los cinco filosofos utilizando 5 tenedores región critica para
--alimentarse bien.

package Filosofo_Tenedor is

   procedure Ocioso;

end Filosofo_Tenedor;
```

3- Codigo de filosofo_tenedor.adb, simple 5 regiones criticas Tenedores y 5 tareas que son los filosofos:

```
with Ada.Real_Time, Ada.Text_IO;
use Ada.Real_Time, Ada.Text_IO;

package body Filosofo_Tenedor is

--tenedor 1 inicia
   protected Tenedor1 is

         pragma priority(2);
      procedure Soltar;
      entry Tomar;
   private
      libre : boolean := true;

   end Tenedor1;
--inicia el body del tenedor
   protected body Tenedor1 is

      procedure Soltar is
      begin
         libre := true;
      end;

      entry Tomar when libre is
      begin
         libre := false;
```

```
    end;
  end Tenedor1;
  --tenedor 1 finaliza

  --tenedor 2 inicia
  protected Tenedor2 is

        pragma priority(2);
    procedure Soltar;
    entry Tomar;
  private
    libre : boolean := true;

  end Tenedor2;
--inicia el body del tenedor
  protected body Tenedor2 is

    procedure Soltar is
    begin
      libre := true;
    end;

    entry Tomar when libre is
    begin
      libre := false;
    end;
  end Tenedor2;
--tenedor 2 finaliza

 --tenedor 3 inicia
  protected Tenedor3 is

        pragma priority(2);
    procedure Soltar;
    entry Tomar;
  private
    libre : boolean := true;

  end Tenedor3;
--inicia el body del tenedor
  protected body Tenedor3 is

    procedure Soltar is
    begin
      libre := true;
    end;

    entry Tomar when libre is
    begin
      libre := false;
    end;
  end Tenedor3;
```

```
--tenedor 3 finaliza

 --tenedor 4 inicia
   protected Tenedor4 is

            pragma priority(2);
      procedure Soltar;
      entry Tomar;
   private
      libre : boolean := true;

   end Tenedor4;
--inicia el body del tenedor
   protected body Tenedor4 is

      procedure Soltar is
      begin
        libre := true;
      end;

      entry Tomar when libre is
      begin
        libre := false;
      end;
   end Tenedor4;
--tenedor 4 finaliza

 --tenedor 5 inicia
   protected Tenedor5 is

            pragma priority(2);
      procedure Soltar;
      entry Tomar;
   private
      libre : boolean := true;

   end Tenedor5;
--inicia el body del tenedor
   protected body Tenedor5 is

      procedure Soltar is
      begin
        libre := true;
      end;

      entry Tomar when libre is
      begin
        libre := false;
      end;
   end Tenedor5;
--tenedor 5 finaliza
```

```
--inicio filosofo1
  task Filosofo1 is
    pragma priority(2);
  end Filosofo1;

  task body Filosofo1 is
    Hora : Time;
    Incremento : Time_Span := Milliseconds(1000);
  begin
    Hora := Clock;
    loop
      Tenedor1.Tomar;
      Put_Line("filosofo1 toma el tenedor1 y medito");
      Hora := Hora + Incremento;
      delay until Hora;
      Tenedor5.Tomar;
      Put_Line("filosofo1 toma el tenedor5 y come");
      Hora := Hora + Incremento;
      delay until Hora;
      Put_Line("filosofo1 suelto tenedor 1 y 5");
      Tenedor1.Soltar;
      Tenedor5.Soltar;
    end loop;
  end Filosofo1;
 -- termino filosofo1

 --inicio filosofo2
  task Filosofo2 is
    pragma priority(2);
  end Filosofo2;

  task body Filosofo2 is
    Hora : Time;
    Incremento : Time_Span := Milliseconds(1000);
  begin
    Hora := Clock;
    loop
      Tenedor1.Tomar;
      Put_Line("filosofo2 toma el tenedor1 y medito");
      Hora := Hora + Incremento;
      delay until Hora;
      Tenedor2.Tomar;
      Put_Line("filosofo2 toma el tenedor2 y come");
      Hora := Hora + Incremento;
      delay until Hora;
      Put_Line("filosofo2 suelto tenedor 1 y 2");
      Tenedor1.Soltar;
      Tenedor2.Soltar;
    end loop;
  end Filosofo2;
 -- termino filosofo2
```

```
 --inicio filosofo3
  task Filosofo3 is
     pragma priority(2);
  end Filosofo3;

  task body Filosofo3 is
     Hora : Time;
     Incremento : Time_Span := Milliseconds(1000);
  begin
     Hora := Clock;
     loop
        Tenedor2.Tomar;
        Put_Line("filosofo3 toma el tenedor2 y medito");
        Hora := Hora + Incremento;
        delay until Hora;
        Tenedor3.Tomar;
        Put_Line("filosofo3 toma el tenedor3 y come");
        Hora := Hora + Incremento;
        delay until Hora;
        Put_Line("filosofo3 suelto tenedor 2 y 3");
        Tenedor2.Soltar;
        Tenedor3.Soltar;
     end loop;
  end Filosofo3;
 -- termino filosofo3

 --inicio filosofo4
  task Filosofo4 is
     pragma priority(2);
  end Filosofo4;

  task body Filosofo4 is
     Hora : Time;
     Incremento : Time_Span := Milliseconds(1000);
  begin
     Hora := Clock;
     loop
        Tenedor3.Tomar;
        Put_Line("filosofo4 toma el tenedor3 y medito");
        Hora := Hora + Incremento;
        delay until Hora;
        Tenedor4.Tomar;
        Put_Line("filosofo4 toma el tenedor4 y come");
        Hora := Hora + Incremento;
        delay until Hora;
        Put_Line("filosofo4 suelto tenedor 3 y 4");
        Tenedor3.Soltar;
        Tenedor4.Soltar;
     end loop;
  end Filosofo4;
 -- termino filosofo4
```

```
 --inicio filosofo5
   task Filosofo5 is
      pragma priority(2);
   end Filosofo5;

   task body Filosofo5 is
      Hora : Time;
      Incremento : Time_Span := Milliseconds(1000);
   begin
      Hora := Clock;
      loop
       Tenedor4.Tomar;
         Put_Line("filosofo5 toma el tenedor4 y medito");
         Hora := Hora + Incremento;
         delay until Hora;
         Tenedor5.Tomar;
         Put_Line("filosofo5 toma el tenedor5 y come");
         Hora := Hora + Incremento;
         delay until Hora;
         Put_Line("filosofo5 suelto tenedor 4 y 5");
         Tenedor4.Soltar;
         Tenedor5.Soltar;
      end loop;
   end Filosofo5;
 -- termino filosofo5


   procedure Ocioso is
   begin
      loop
         null;
      end loop;
   end Ocioso;

end Filosofo_Tenedor;
```