Sistemas en tiempo real

Alumno: Diego Díaz Alvarellos
Curso 2015-2016
PED 3.

Contenido

Enunciado.

Un sistema operativo concreto tiene una llamada de sistema (EjecutarConcurrentemente) que admite un array ilimitado. Cada elemento del array es un apuntador a un procedimiento sin parámetros. La llamada de sistema ejecuta concurrentemente todos los procedimientos, y devuelve control cuando todos han finalizado. Muestre cómo puede implementarse en Ada esta llamada de sistema utilizando las funcionalidades de tareas de Ada. Suponga que el sistema operativo y la aplicación se ejecutan en el mismo espacio de direcciones.

Solución

Se ha tomado la solución del libro, explicado y probado el código. Se entrega junto a este documento, un archivo comprimido que contiene el código fuente (archivo concurrente.adb) junto a los archivos generados en el proceso de compilación.

Explicación del código

En la cabecera del programa principal importa la librería Text_IO.

```
-- Cláusula de contexto: uso de la librería "Text_IO"
-- Proporciona servicios básicos de E/S
with Ada.Text_IO; use Ada.Text_IO;

procedure Principal is
```

Se definen los Procedimento, cada elemento del array es un apuntador a uno de ellos. Se retarda la llamada de escritura de los procedimientos uno y dos, de esa forma se evita que se muestren los mensajes al mismo tiempo.

```
procedure ProcedimientoUno is
begin
    Put_Line("Procedimiento Uno Finalizado!!");
end ProcedimientoUno;

--------------------------------------------------------------------------------

procedure ProcedimientoDos is
begin
    delay 1.0;
    Put_Line("Procedimiento Dos Finalizado!!");
end ProcedimientoDos;

--------------------------------------------------------------------------------

procedure ProcedimientoTres is
begin
    delay 2.0;
    Put_Line("Procedimiento Tres Finalizado!!");
end ProcedimientoTres;
```

Definición de los tipos puntero y array de punteros.

```
--TipoPuntero: permite llamar a un subprograma sin conocer su nombre ni su ubicación
type TipoPuntero is access procedure;
-- TipoParametros: array no restingido de tipo TipoPuntero
type TipoParametros is array(Positive range <>) of TipoPuntero;
```

EjecutarConcurrentemente: recibe como argumento un tipo TipoParametros (array ilimitado del tipo TipoPuntero)
Tarea Principal: recibe como argumentos un TipoPuntero (permite llamar a un subprograma sin conocer su nombre ni su ubicación)

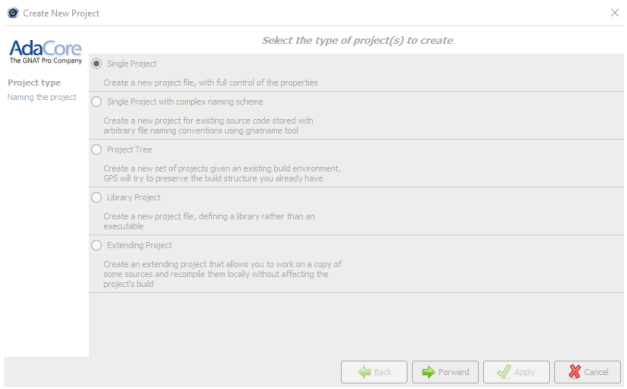
```
procedure EjecutarConcurentemente (Entrada : TipoParametros) is
task type TareaPrincipal(Puntero: TipoPuntero);
task body TareaPrincipal is
begin
    Puntero.all;
end TareaPrincipal;

-- Tipo de los punteros a TareaPrincipal dinámicos
type TareaPrincipal_TipoPuntero is access TareaPrincipal;
Starter : TareaPrincipal_TipoPuntero; --variable estatica de tipo TareaPrincipal

begin -- Cuerpo EjecutarConcurrentemente
    Put_Line("EjecutarConcurrentemente Inicializada");
    -- Iteración desde el límite inferior del Array Entrada (que pasado como argumento) hasta su límite superior
    for i in Entrada'Range loop
    -- Variable dinámica de tipo Worker, inicializada con el valor de la iteración contenido en Entrada
    Starter := new TareaPrincipal(Entrada(i));
    end loop;
    Put_Line("EjecutarConcurrentemente Finalizada");
end EjecutarConcurentemente;
```

Inicio del programa principal

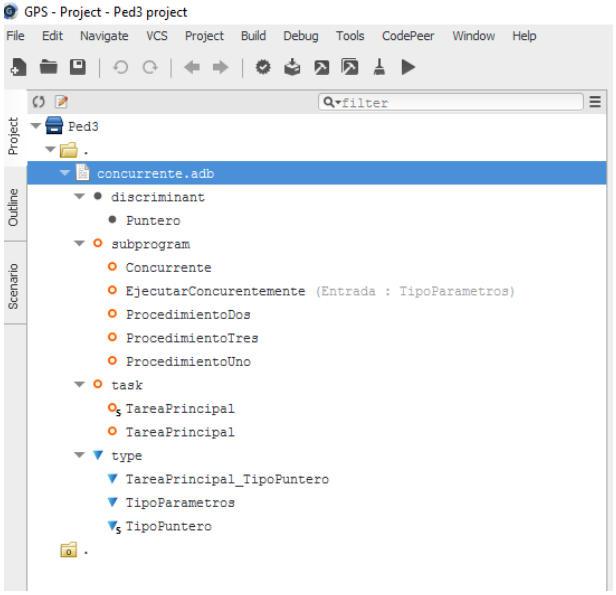
```
begin
Put_Line("Iniciado Programa Principal");
EjecutarConcurentemente((ProcedimientoUno'access, ProcedimientoDos'access, ProcedimientoTres'Access)) ;
Put_Line("Programa Principal Finalizado");
end Concurrente;
```

Test de ejecución

PASO 1 : Se instala el entorno GPS. AdaCore (Gnat Pro Company)
PASO 2 : Se crea un nuevo proyecto.



PASO 3. Compilación, construcción.



PASO 4. Ejecución del programa