Tareas en Ada

Lecturas recomendadas:
A. Burns, A. Wellings, sección 4.1
Cohen, capítulo 17
Ada Reference Manual, Section 9

Manuel Carro

Universidad Politécnica de Madrid

6 de octubre de 2008

Tareas en Ada

- Anónimas (sólo una de cada tipo)
- Como tipos:
  - Sin paso de parámetros
  - Con paso de parámetros (tipos paramétricos)
- Una declaración de variable **lanza** la tarea
- Independientemente: punteros a tareas
  ⟶ arranque dinámico

Arranque de tareas en Ada

Tareas anónimas

```
bio.adb, andes{1,2,3,4,5}.adb
task Tarea_A;

task body Tarea_A is
begin
  -- <código tarea, como un procedimiento>
end Tarea_A;
```

- Sólo una instancia de la tarea

Arranque de tareas en Ada (Cont.)

Tareas como tipos

simult.adb

```
task type Tipo_A;

task body Tipo_A is
begin
  -- <código tarea, como un procedimiento>
end Tipo_A;
```

- Declaración de variable crea tarea (*apuntada* por variable)

  ```
  Tarea_A: Tipo_A;
  ```

- En cierto sentido: variable ≡ tarea
- Varias tareas con código idéntico

  ```
  Varias_Tareas_A: array (1..NA) of Tipo_A;
  ```


Arranque de tareas en Ada (Cont.)

Paso de parámetros iniciales

simult1.adb

```
task type Tipo_A(Parametro: Natural);

task body Tipo_A is
begin
  -- <código tarea>
  Put(Parametro);  -- Por ejemplo...
end Tipo_A;
```

- Restricciones sobre tipo de Parametro
- Tareas con diferentes paramétros iniciales

  ```
  Tarea_1: Tipo_A(1); -- Arranca una tarea
  Tarea_2: Tipo_A(2); -- Arranca otra tarea
  ```


Arranque de tareas en Ada (Cont.)

Creación dinámica de tareas

simult2.adb

```
task type Tipo_A(Parametro: Natural);

task body Tipo_A is
begin
  -- <código tarea>
  Put(Parametro);  -- Por ejemplo...
end Tipo_A;
```

- **Idea:** crear la variable cuando sea necesario

  ```
  type Tipo_Punt_A is access Tipo_A;
  Una_Tarea: Tipo_Punt_A;

  begin
    Una_Tarea := new Tipo_A(3);
  end ...;
  ```

Excepciones en una tarea

- **Las excepciones en Ada no se propagan fuera de las tareas** (`excep.adb`)
- ¡Resultados inesperados!

```
task body Inutil is
begin
   loop
      delay 10.0;
   end loop;
end Inutil;
```

```
task body Erronea is
   I: Natural := 1;
begin
   delay 1.0;
   I := 1 / (I-I);
end Erronea;
```

- Programa no termina
- **Capturar excepciones en las tareas**