Paquetes, Algoritmos de Planificación, Protocolos de Sincronización

(p. 30, 44/232 Tesis MAR)

Los paquetes constituyen unidades de compilación independiente. Esto facilita el desarrollo grandes aplicaciones, puesto que los módulos pueden ser compilados y probados de forma independiente por los distintos equipos de trabajo sin necesidad de disponer aun de la aplicación final. Además la estructura del lenguaje permite conocer en tiempo de compilación las dependencias entre los distintos módulos sin que sea necesaria la intervención del programador como ocurre en otros lenguajes (como el C y el uso de los “makefiles”). Esto permite automatizar el proceso de compilación de grandes aplicaciones, con lo que se puede optimizar el número de recompilaciones a realizar tras una modificación y se evitan los errores que la intervención del programador puede provocar.

Una de las características más relevantes del lenguaje Ada es la fiabilidad del código generado, conseguida gracias a que las reglas del lenguaje permiten detectar numerosos errores en tiempo de compilación. Así, entre otras situaciones, se producirá un error de compilación cuando no se respete el modo (entrada, salida o entrada-salida) de un parámetro de un procedimiento o cuando no se hayan considerado todas las alternativas en una sentencia condicional múltiple (case). Pero la mayor parte de los errores detectados durante la fase de compilación lo son gracias a la propiedad de tipología estricta del lenguaje. Dicha propiedad obliga al programador a hacer explícita cualquier conversión de tipos que desee realizar, suponiendo un error de compilación la utilización de un tipo indebido en una expresión, como un parámetro de un procedimiento o función o en cualquier otra construcción del lenguaje.

// …

Otra característica destacable la constituye el hecho de que la concurrencia esté directamente soportada por el lenguaje, existiendo primitivas que permiten definir las actividades concurrentes, denominadas tareas, así como sus interacciones. La planificación de las tareas se realiza mediante una política expulsora basada en prioridades con orden FIFO dentro de la misma prioridad, como la definida en el apartado 1.2 “Algoritmos de planificación”. Se definen dos formas de interacción entre tareas: el envío de mensajes o “rendezvous” y la utilización de objetos protegidos. Estos últimos permiten el acceso exclusivo a datos compartidos mediante la utilización del protocolo de techo prioridad inmediato, también denominado protección de prioridad que fue explicado en el apartado 1.3, “Protocolos de sincronización”. Tanto para la política, como para los mecanismos de sincronización elegidos se dispone de una amplia base teórica, lo que permite programar aplicaciones con tiempos de respuesta predecibles.

Una de las más importantes aportaciones del Ada 95 frente a su antecesor Ada 83 es que soporta de forma completa el paradigma de programación orientada a objetos. Además del encapsulamiento y ocultación de información permitido por los paquetes (ya disponible en Ada 83), en Ada 95 se definen los registros etiquetados (tagged). Este nuevo tipo de dato permite aplicar los conceptos de herencia y polimorfismo, que son básicos en la programación orientada a objetos.