RMI-C++:

Un Paquete para la invocación de métodos remotos en C++

El proyecto denominado *RMI-C++: Un Paquete para la invocación de métodos remotos en C++*, tiene como objetivo el desarrollo de un paquete de clases junto con un compilador que permita el uso e invocación de objetos remotos en C++.

El desarrollo de aplicaciones cliente/servidor usando *sockets* conlleva el diseño de un protocolo consistente en un lenguaje común entre el cliente y el servidor. El diseño de dicho protocolo no siempre es sencillo y es una fuente de errores tales como el *deadlock*.

En vez de trabajar directamente con *sockets*, las aplicaciones cliente/servidor, pueden ser desarrolladas mediante la invocación de métodos de objetos que están ejecutándose en otra máquina virtual (posiblemente en otro host). RMI-C++ intenta mejorar estos aspectos, mediante el desarrollo de un conjunto de librerías de clases en C++ y un compilador que genere el código necesario para la creación y comunicación de objetos.

Objetivos

Los objetivos de RMI-C++, por tanto, serán son los siguientes:

- Soporte de invocación de objetos remotos en diferentes máquinas virtuales.
- Soporte de retorno de resultados desde el servidor a los clientes.
- Integrar el modelo de objetos distribuidos en el lenguaje de una forma natural, mientras se mantiene la semántica de objetos ya definidas.
- Realizar aplicaciones distribuidas lo más simples y legibles posibles.

Etapas de Desarrollo

El proyecto se llevará a cabo siguiendo 5 etapas:

1. Estudio de la comunicación vía sockets en UNIX.
2. Implementación de un conjunto de clases de comunicación entre objetos sobre TCP/IP.
3. Desarrollo manual de las clases Stub y Skeletons que proporcionen el interfaz transparente de comunicación remota.

4. Desarrollo e implementación de la herramienta o compilador que genere de manera automática el código necesario para el enlace y/o creación con objetos remotos, así como su comunicación.
5. Pruebas y depuración del producto final.

Medios Materiales

El proyecto será desarrollado bajo UNIX, en lenguaje C++ junto con el uso de las herramientas lex y *yacc* para el análisis y generación de código y junto con las librerías de sockets del sistema..

- Compilador de *C++*.
- Entorno de Red (PC's, estaciones de trabajo, etc.)
- Lex y Yacc

Referencias

- *Remote Method Invocation System*. Sun Microsystem Inc.
- *Compiladores : principios, técnicas y herramientas*. Alfred V. Aho, Ravi Sethi.
- *Introduction to compiling techniques : a first course using ANSI C, LEX and YACC* . J.P. Bennet.
- Lex & yacc. John R. Levine, Tony Mason, Doug Brown.
- Bertrand Meyer. Object-Oriented Software Construction, Prentice Hall, 1988.
- Bertrand Meyer. Systematic Concurrent Object-Oriented Programming, Communications of the ACM, vol. 36, no. 9, September 1993.
- Kim Waldén and Jean-Marc Nerson. Seamless Object-Oriented Software Architecture: Analysis and Design of Reliable Systems, Prentice Hall, 1995.
- M. Raynal. Distributed Algorithms and Protocols, John Wiley & Sons Ltd., 1988.
- R. Andrews. Concurrent Programming, Gregory Benjamin/Cummings, 1991.
- Schildt, Herbert. C ++ : Manual de referencia / Herbert Schildt. 1999
- Kate Gregory. Microsoft Visual C++ 6, Prentice Hall, 1988