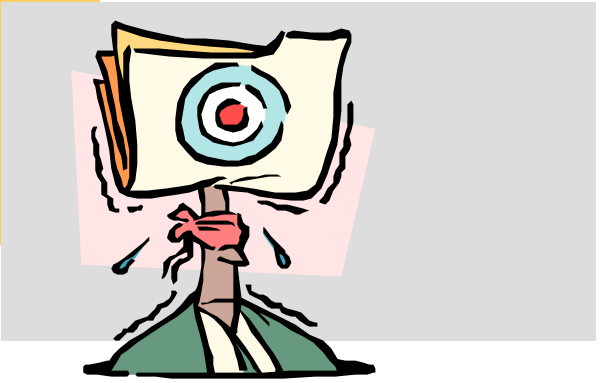
Facultad de Informática UCM
JPM, 1999-2000
Diseño de sistemas operativos

Una aplicación sencilla con C++ y CORBA

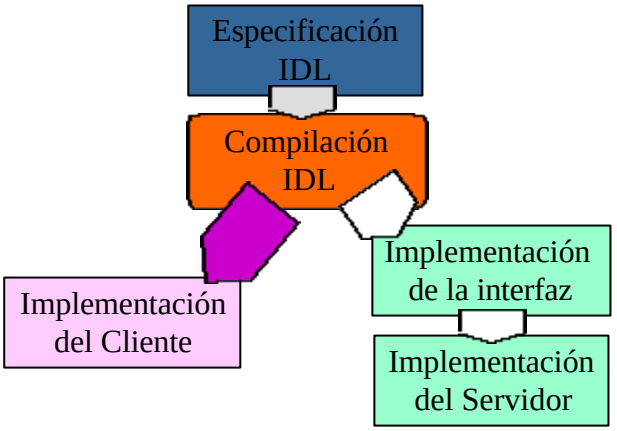


Contenidos

Tema 3: Una aplicación sencilla con C++ y CORBA

- Descripción de la aplicación
- Definición de interfaz con IDL
- Compilación de la interfaz IDL
- Programación del Cliente
- Programación de la implementación de interfaz y del servidor
- Compilación y Ejecución
- Resumen

Construcción de la aplicación CORBA

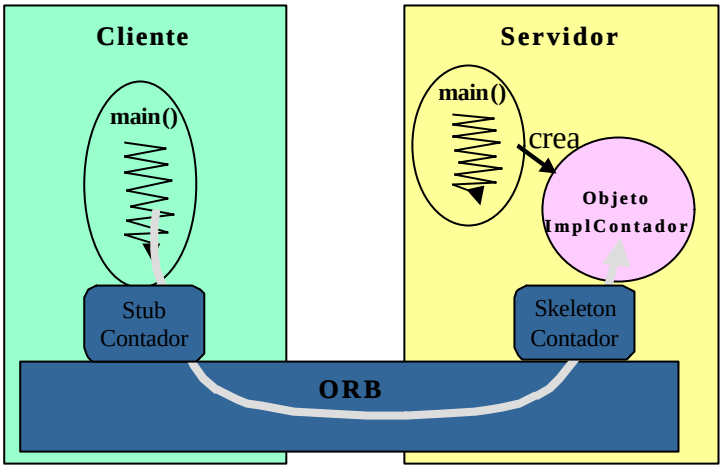


Descripción de la aplicación

Se trata de implementar un programa cliente/servidor muy sencillo: un Contador distribuido.

- Los clientes pueden consultar y modificar el valor del contador. También hay operaciones para incrementar y decrementar en una unidad el valor del contador.
- La aplicación muestra el modo de invocación estático (stubs y skeletons generados por el compilador IDL)
- El programa cliente invoca 1000 veces la operación incrementar sobre el contador y al final muestra el tiempo medio de respuesta

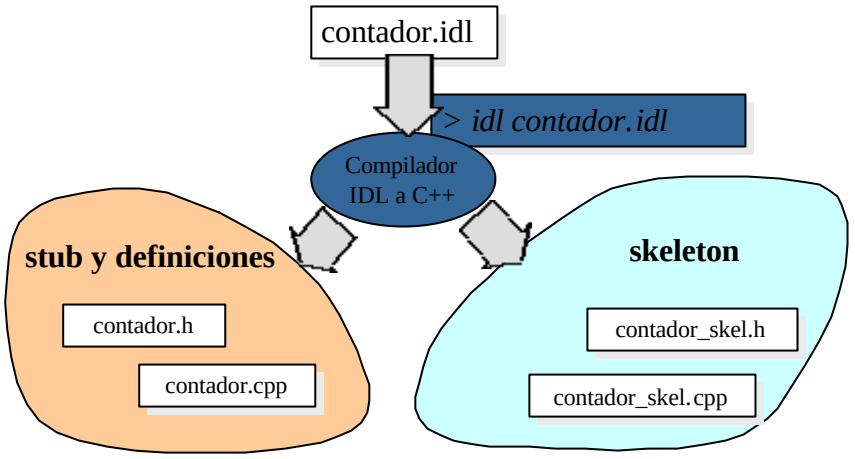
Arquitectura de la aplicación



Especificación de la interfaz con IDL

```
// contador.idl

module EjemploContador {

  interface Contador {
        attribute long valor;

        long inc();
        long dec();
  };

};
```

Compilación de la interfaz



(Ficheros generados con el compilador IDL de ORBacus)

Programación del Cliente

- El Cliente puede ser simplemente una función **main()**
- En general un Cliente realiza las siguientes funciones:
  1) Inicializar el ORB
  2) Obtener la referencia a un objeto CORBA que implemente la interfaz deseada (**en este ejemplo, *Contador***)
     - Usando la función del ORB *string_to_object()*, si se consigue la referencia como una cadena de caracteres (por ejemplo de un fichero)
     - Usando el servicio de Nombres (mejor)
  3) Utilizar el objeto CORBA

ClienteContador.cpp

```
#include <OB/CORBA.h>
#include <contador.h>

#include <fstream.h>
#include <time.h>
#include <sys\types.h>
#include <sys\timeb.h>

int main (int argc, char* argv[], char*[]) {
   try {
      // 1. Inicializa el ORB
            CORBA_ORB_var orb = CORBA_ORB_init(argc, argv);

      // 2. Obtiene la referencia al objeto Contador
            const char* fichero = "Contador.ref";
            ifstream entrada(fichero);
            char s[100];
            entrada >> s;
            CORBA_Object_var objeto = orb->string_to_object(s);
            Contador_var c = Contador::_narrow(objeto);
            entrada.close();
   // ...
```


ClienteContador.cpp

```
   // 3. Utiliza el contador:
         cout << "El Contador 1 tiene el valor: " << c->valor() << endl;

         cout << "Incrementando el contador..." << endl;

         struct timeb tiempo;
         ftime(&tiempo);
         double t_inicial = ((double)tiempo.time+((double)tiempo.millitm)/(double)1000);

         for (int i = 0 ; i < 1000 ; i++ )   c->inc();

         ftime(&tiempo);
         double t_inicial = ((double)tiempo.time+((double)tiempo.millitm)/(double)1000);

         cout << "Tiempo medio de invocacion = ";
         cout << (t_final - t_inicial) << " msegs" << endl;
         cout << "Valor final del contador = " << c->valor() << endl;
   }
   catch (CORBA::SystemException& e) {
         cout << "System Exception: " << e << endl;
         return(1);
   }
}
```

Implementación del servidor

El servidor consta de dos partes:

1) La implementación de una o varias interfaces IDL
   - Clases que implementan las operaciones de las interfaces y que se corresponden a objetos CORBA
   - Hay dos maneras de implementar una interfaz:
     - Por herencia de la clase que implementa el skeleton *X_skel*
     - Por delegación, usando la clase *_tie_X*
2) El programa principal (***main***)
   - Inicializa el ORB
   - Crea objetos que implementan interfaces

Implementación de la interfaz

- En este ejemplo se utiliza la herencia de la clase que implementa el skeleton
- La implementación se realiza definiendo una clase de implementación que tiene que hacer 2 cosas:
  1) Heredar de la clase *<interfazIDL>_skel*
     - Esta clase la genera el compilador de IDL y tiene los métodos que implementan el skeleton
  2) Implementar cada método de la interfaz
     - Se pueden usar métodos auxiliares en la clase de implementación de la interfaz

Implementación de la interfaz Contador ImplContador.h

```
#include <contador_skel.h>

class ImplContador : public contador_skel {
  private:
    long valor_;

  public:
   ImplContador (long valorinicial);

  // attribute valor
   virtual long valor();
   virtual void valor(long v);

  // operaciones de la interfaz IDL:
   virtual long inc();
   virtual long dec();
};
```


Implementación de la interfaz Contador ImplContador.cpp

```
#include <OB/CORBA.h>
#include <ImplContador.h>

ImplContador::ImplContador (long valorinicial) {
    valor_ = valorinicial;
  }

long ImplContador::valor() { return valor_;  }
void ImplContador::valor(long v) { valor_ = v; }

  // operaciones:
long ImplContador::inc() { return ++valor_; }
long ImplContador::dec() { return --valor_; }
```

Implementación del servidor
Programa principal (main)

El programa principal (**main**) realiza las siguientes funciones:

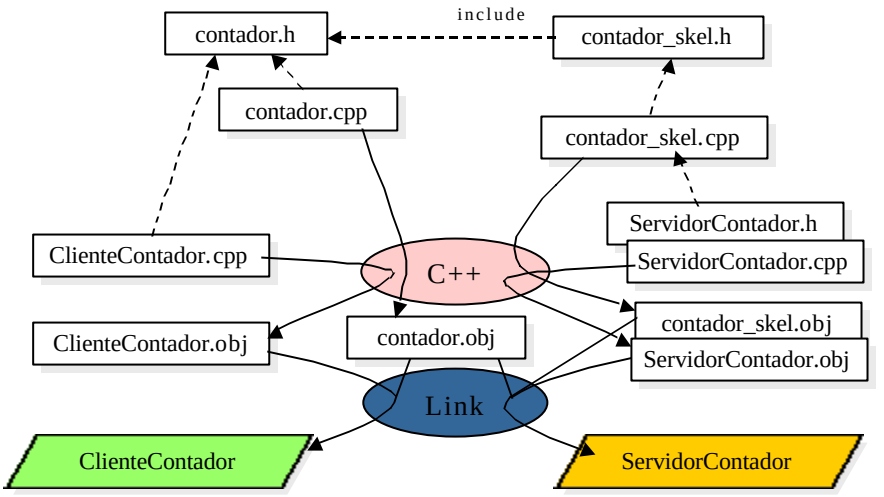
1) Inicializar el ORB
2) Crear los objetos CORBA
   - Al menos los que sean necesarios inicialmente (se pueden crear otros dinámicamente)
3) Pasar el control al ORB
   - Bucle de eventos

Implementación del servidor
ServidorContador.java

```
#include <OB/CORBA.h>
#include <ImplContador.h>

#include <fstream.h>

int main(int argc, char* argv[], char*[]) {
  // Inicializa el ORB
    CORBA_ORB_var orb = CORBA_ORB_init(argc, argv);
    CORBA_BOA_var boa  = orb-> BOA_init(argc, argv);

  // Crea el objeto Contador
    Contador_var c = new ImplContador(0);

    CORBA_String_var s = orb->object_to_string(c);
    const char* fichero = "Contador.ref";
    ofstream salida(fichero);
    salida << s << endl;
    salida.close();

  // Listo para recibir peticiones
    boa->impl_is_ready(CORBA_ImplementationDef::_nil());
}
```

©

Compilación de la aplicación



Prueba de la aplicación

Una vez compilados cliente y servidor, basta con ejecutarlos (primero el servidor):

```
> ServidorContador &
> ClienteContador
Inicializa el ORB...
Consigue la referencia al objeto Contador1
El objeto Contador1 tiene el valor: 0
Incrementando el contador...
Tiempo medio de invocacion = 3.02 msegs
Valor final del contador = 1000
```

Resumen

- La realización de una aplicación cliente/servidor con C++ y CORBA requiere:
  - Definir con IDL las interfaces que exporta el servidor
  - Desarrollar el cliente usando los stubs generados por el compilador de IDL
  - Implementar las interfaces y desarrollar el servidor utilizando los skeletons generados por el compilador de IDL