DC ASSIGNMENT – RPCGEN files generated

SIDDHARTHA BANERJEE 2017HT13125

What are the purpose of the extra files which gets generated after running the "rpcgen" program? [2 mark]

The extra files that are generated after running the "RPCGEN" program are:

File name generated	File Type	Description
sidd_factorial.h	RPC_HDR	Header File Output
sidd_factorial_clnt.c	RPC_CLNT	Client Tub Output
sidd_factorial_svc.c	RPC_SVC	Server Stub Output
sidd_factorial_xdr.c	RPC_XDR	XDR Routine Output

a) **Header:**

A header file of definitions common to the server and the client. The header file contains definitions of all data types and operations declared in the IDL file, as well as all data types and operations declared in the files included with the #include directive.

Data types from the files imported with the import directive are not replicated to the header file; instead the generated header file contains an include line to the H file generated from the imported file.

The header file must be included by all application modules that call the defined operations, implement the defined operations, or manipulate the defined types

b) **Stub File:**

A stub in distributed computing is a piece of code that converts parameters passed between client and server during a remote procedure call (RPC).

The main idea of an RPC is to allow a **local computer (client) to remotely call procedures on a different computer (server)**. The client and server use different address spaces, so parameters used in a function (procedure) call have to be converted, otherwise the values of those parameters could not be used, because pointers to parameters in one computer's memory would point to different data on the other computer.

The client and server may also use different data representations, even for simple parameters (e.g., big-endian versus little-endian for integers). **Stubs perform the conversion of the parameters, so a remote procedure call looks like a local function call for the remote computer**. Stub libraries must be installed on both the client and server side.

i) <u>Client Side Stub:</u>

A client stub is responsible for conversion (marshalling) of parameters used in a function call and deconversion of results passed from the server after execution of the function.

The client stub module provides surrogate entry points on the client for each of the operations defined in the input IDL file.

When the client application makes a call to the remote procedure, its call first goes to the surrogate routine in the client stub file. The client stub routine performs the following functions:

- Marshals arguments. The client stub packages input arguments into a form that can be transmitted to the server.
- Calls the client run-time library to transmit arguments to the remote address space and invokes the remote procedure in the server address space.
- Unmarshals output arguments. The client stub unpacks output arguments and returns to the caller.

ii) <u>Server Side Stub:</u>

A server skeleton, the stub on the server side, is responsible for deconversion of parameters passed by the client and conversion of the results after the execution of the function.

The server stub provides surrogate entry points on the server for each of the operations defined in the input IDL file.

When a server stub routine is invoked by the RPC run-time library, it performs the following functions:

- Unmarshals input arguments (unpacks the arguments from their transmitted formats).
- Calls the actual implementation of the procedure on the server.
- Marshals output arguments (packages the arguments into the transmitted forms).

c) XDR Routine File:

External data representation (XDR) is a standard for the description and encoding of data. The XDR protocol is useful for transferring data between different computer architectures and has been used to communicate data between very diverse machines.

It allows data to be transferred between different kinds of computer systems. Converting from the local representation to XDR is called encoding. Converting from XDR to the local representation is called decoding.

XDR is implemented as a software library of functions which is portable between different operating systems and is also independent of the transport layer. XDR fits into the ISO reference model's presentation layer (layer 6).

XDR uses a language to describe data formats and can only be used to describe data. It is not a programming language. This language enables you to describe intricate data formats in a concise manner. The XDR language is similar to the C language.