This document contains information about the server side of the steering library and assumes familiarity with the mpi_ message passing calls of the Paragon. The Library Calls on the server side are :

G_InitGraphicsNode
G_SetVariables
G_SetParameters
G_Run
G_TerminateConnection

G_InitGraphicsNode

Initialize Graphics Node.
Set up the transport protocol.

SYNOPSIS

int G_InitGraphicsNode (int GNode, int nCNode, int * nCNode);

DESCRIPTION

This function call sets up the transport protocol on the graphics node. For RPC to succeed a *daemon* is set up on the graphics node. Thus the graphics node never returns from this call.

Note that the syntax of this call is slightly different from that of the nx version. In the nx version, the controller acts as the compression node. The intent here is to have future versions of the MPI version execute our parallel compression algorithms in a compute group. Currently the parameters passed to this call are ignored.

PARAMETER

RETURN VALUE :

0 on success.
$Non - Zero$ on failure

G_SetVariables

The compute nodes register their respective variables.

SYNOPSIS

int G_SetVariables (VariableNames, VariableType, VariableSize)
char * ArrayOfVariableNames,
int ArrayOfVariableType,
int ArrayOfVariableSize

FUNCTION

Before stating the functionality, two assumptions have been made here:

1. The host can only request those variable names, that have been registered here.
2. None of these arrays ever in their lifetime will exceed the size specified in VariableSize.

This function is called by the compute program immediately after initialization of the graphics node. Once these variables have been registered, the Server starts listening for server requests and not before.

This function returns a message type which can then be used to communicate with the graphics node.

A major point to be noted here is that the RPC server is designed to run on the controller node which is in a logically different communication group than the group that the compute nodes are in. This group is automatically created by the library.

PARAMETERS

*char * ArrayOfVariableNames*
This is an array of strings which contains the names of the variables being registered.
Each of these names can be maximum 80 characters in length.

*int * ArrayOfVariableType*
This specifies the type of these variables. Instead of passing actual type names like *float*, *doubles*, the expected values are the codes as specified in the "graphics.h" file.
This sort of scheme will assist in incorporating structures.

*int * ArrayOfVariableSize*
This array specifies the maximum size that any of the arrays may have. This value should not exceed the value MAX_VAR_SIZE as listed in the graphics.h file.

RETURN VALUE

Returns a message type if successful on registration, -1 on failure

FILES

server/server_mpi.c Source.

G_SetParameters

Called by compute nodes to register the parameters.

SYNOPSIS

int G_SetParameters (VariableNames, VariableType, VariableSize)
char * ArrayOfVariableNames;
int ArrayOfVariableType;
int ArrayOfVariableSize;

DESCRIPTION

Before stating the functionality, we state two assumptions here. The previous function **G_SetVariables** registered variables, i.e. data which is passed from the server to the client. Parameters are data which are passed from client to server in order to steer an application.

1. The server can only send those parameter names, that have been registered here. Each of these named parameters is an array of a specific type.

2. None of these arrays ever in their lifetime will exceed the size specified in VariableSize.

This function is called by the compute program immediately after initialization of the graphics node.

The return value is a message type which can be used for communication with the graphics node.

A major point to be noted here is that the RPC server is designed to run on the controller node. Hence the user program should have the mpi_initve and mpi_fork style of startup code. This function should be called by the parent of mpi_fork (i.e. by the controller process).

PARAMETERS

1. The first variable is a string which contains the name of the parameters being registered. Each of these names can be maximum 80 characters in length.

2. This specifies the type of the variable. Instead of passing actual type names like "float", "doubles", the expected values are the codes as specified in the "graphics.h" file. This sort of scheme will assist in incorporating structures.

3. This array specifies the maximum size that any of the arrays may have. This value should never exceed MAX_VAR_SIZE as specified in the graphics.h file.

RETURN VALUE

$MessageType$: success
-1 : Insufficient memory

G_Run

SYNOPSIS

This function initiates the server to start listening to requests from the clients.

void G_Run();

DESCRIPTION

This function initiates the server to start listening to requests from the clients and should only be called only once by one of the compute nodes.

G_TerminateConnection

SYNOPSIS

Terminate the connection from the server side.

void G_TerminateConnection();

DESCRIPTION

Terminate the connection for the server side and free up any memory allocated during previous calls to G_SetVariables or G_SetParameters.

FUNCTION PARAMETERS

RETURN VALUES

FILES

server/server_mpi.c Source.

TIPS

BUGS

SEE ALSO

G_InitGraphicsNode on server side
FAQ

BRIEF NOTE ON MESSAGE PASSING

A compute node must communicate with the graphics processor. This communication can be done either synchronously or asynchronously. The message types to be used during communication are the message type returned by the Register calls.