15640 Project 1: Transparent Remote File Operations

The design of project 1 in my file includes: Client marshall data and send to server; Server receive massage from client, unmarshall the data and execute the file operations; Server marshall return content, send back to client; Client unmarshall the returned data and return demanded value to user.

The details of the design are listed below,

1. Data Marshalling
   1) Client: First distinguish what the function call is. Use a header to contain the name of the function call. Since the length of different function calls are not the same, define HEAD_LEN = 16, put the string of call name in the first several bytes. The following bytes are all set to '\0'.
      Second, calculate the number of bytes that will be used to contain all the input parameters of function call. Use next 4 bytes to contain the number.
      Third, combine all the input parameters into a string. (Keep the null terminator '\0' of input strings)
      Last, combine all three-above parts into one single string.
   2) Server: First use demanded bytes to contain the return value.
      Second, Use next 4 bytes to contain errno.
      Third, for function call like: read, stat, getdirentries, we need to contain other demanded return contents. For read, read buffer content needs to be sent back. For stat, stat_buf content is needed. For getdirentries, both the getdirentries buf content and off_t pointer basep content are demanded.
      Last, combine all three-above parts into a single string.
   3) Special Case: getdirtree: server call getdirtree and the return value is a dirtreenode address. Serialize the root dirtreenode to a string, then combine the length of serialized string, errno and serialized string in one string and send back to server.
      freedirtree: solved locally on the client side. No need to send to server.

2. Send and Recv between Clients and Server
   1) Client: Send the marshaled data to server. (Send once)
      Server: Receive twice. First receive a length of HEAD_LEN + sizeof(int) data. Use strcpy() to get the name of function call. Seperate last 4 bytes into the length of message (msg_len) to be received next. Second, use the name of function call to direct to different functions. Receive msg_len bytes data. Seperate the data into different parameters. Use these parameters as inputs to call different functions.
   2) Server: Marshal return value, errno and demanded return content. Send back. (Send once)
      Client: For most function calls, receive once, separate return value and errno, and copy content to buffer address.
      For read, getdirtree operation, we need to receive twice. For read, first receive the return value and errno. Then use return value as receive length to receive read buffer content. For getdirtree, first receive the length of serialized string and errno. Then use the length to receive serialized string, deserialize it to dirtree and return the address of root dirtreenode.

   Build wrapper functions for send(), recv(). Use loop to send or receive the demanded length of message.