Subject: Re: Convert struct to string and reconstruct a struct from string
Posted by mirek on Fri, 30 Oct 2020 08:33:02 GMT
View Forum Message <> Reply to Message

sinpeople wrote on Fri, 30 October 2020 02:35Hi folks,

I have a client and server which communicates via UDP. Ideally, the client converts one struct to strings and sent it to server with its client id and a command id. The message would be like "ClientID, MessageID, Strings converted from a struct". The server side picks up the message and from the messageID, it knows which struct to be used to recover its content from the remaining portion of the message, at the server side.

In case that I have roughly about 100+ such structs, how do I construct this portion to avoid a huge switch case to make the program lean with current available resources in U++?

Please point me to the right direction. I am very new to U++.

Thank you very much!

Best Regards
David

As this sounds like you have both server and clinet under your control, I would say that binary serialization here makes the sense.

One question that remains is about what you are going to do with that struct then...

Either way, I think where this leads is that you will have Serialize method in all of your structs. While it is probably not only way how to do things, I think that in this case it will be reasonable to have some base class for your structs and Serialize will then be virtual.

```
struct AMessage {
  virtual void Serialize(Stream& s) = 0;
};

struct TemperatureMessage {
  double altitude, temperature;

  virtual void Serialize(Stream& s) {
     s % altitude % temperature;
  }
}
```

Then I can imagine you will have a map somewhere to create the specific struct on demand:

```
VectorMap<int, void (*make)(One<AMessage>& m)> message_maker;

template <class T>
void RegisterMessage(int messageid)
{
    message_maker.Add(messageid, [](One<AMessage>& m) { m.Create<T>(); });
}

INITBLOCK {
  RegisterMessage<TemperatureMessage>(); // do that for all of your messages
};
```

then when processing the intput

```
void ProcessRequest(const String& data)
{
  StringStream ss(data); // error handling for now omitted
  int client_id = ss.GetInt32();
  int message_id = ss.GetInt32();
  One<AMessage> m;
  int q = message_maker.Find(message_id);
  if(q < 0)
    return;
  (*message_maker[q])(m); // create the required concrete message
  ss % *m; // load data to struct
}
```

Of course, this all is based on very little info that you have provided...

Mirek
