Subject: error: call to implicitly-deleted copy constructor
Posted by sinpeople on Tue, 23 Feb 2021 03:29:51 GMT
View Forum Message <> Reply to Message

Hi folks,

Newbie here.

I got this "call to implicitly-deleted copy constructor" error in this very simple program.

The error changes to "no matching constructor for initialization of xxx" after I added a copy constructor into the class. Adding a default constructor explicitly into the code didn't get rid of this new error.

The code is attached with the copy constructor and default constructor in comment. If the .push_back statement is commented, no error will show up. I guess it has something to do with the containers, which i am not familiar at all.

Please help me. Thank you very much!

Best Regards
David
 ========== ===

```
#include <Core/Core.h>
#include <deque>

using namespace Upp;
using namespace std;

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

struct SignalStatus : Moveable<SignalStatus> {
 int sg_ID;
 int sg_Color;
 int sg_Elapsed;
    virtual void Serialize(Stream& s) {
       s % sg_ID % sg_Color % sg_Elapsed;
    }
};

struct SignalStatusMessage : AMessage {
```

```
 int nTime;
 Vector<SignalStatus> status;
/*
    SignalStatusMessage(){};
    SignalStatusMessage(SignalStatusMessage& temp)
    {
        nTime = temp.nTime;
        status.clear();
        for(int i=0; i<temp.status.size(); i++)
        {
            SignalStatus sgStatus;
            sgStatus.sg_ID = temp.status[i].sg_ID;
            sgStatus.sg_Color = temp.status[i].sg_Color;
            sgStatus.sg_Elapsed = temp.status[i].sg_Elapsed;
         status.push_back(sgStatus);
        }
    }
*/
    virtual void Serialize(Stream& s) {
       s % nTime % status;
    }
};


CONSOLE_APP_MAIN
{
 //deque<String> deqAbs;
 SignalStatusMessage sigStatus;
 deque<SignalStatusMessage> deqSigStatus;

 deqSigStatus.push_back(sigStatus);
}
```

Subject: Re: error: call to implicitly-deleted copy constructor
Posted by peterh on Tue, 23 Feb 2021 13:45:09 GMT
View Forum Message <> Reply to Message

There where changes in the U++ Framework last year, to avoid unnecessary copies of objects.
The compile error is fixed if you change line 53:

```
 deqSigStatus.push_back(pick(sigStatus));
```

See here:
https://www.ultimatepp.org/srcdoc$Core$pick__en-us.html

I am not sure if this the proper solution, because I am not so skilled with U++

Subject: Re: error: call to implicitly-deleted copy constructor
Posted by Novo on Tue, 23 Feb 2021 16:55:52 GMT
View Forum Message <> Reply to Message

STL is designed the way that it silently makes copies of objects for you.
U++ is preventing this behavior. You have to explicitly choose to copy/clone or to move/pick your object.
This requires a little bit more coding, but resulting code is more efficient.

Subject: Re: error: call to implicitly-deleted copy constructor
Posted by peterh on Tue, 23 Feb 2021 18:44:10 GMT
View Forum Message <> Reply to Message

Finding the problem was very hard for me some months ago, because I am not very skilled and have no experience and only basic knowlege about STL.

So an explanation how to locate this error:
In the Message list there is one error displayed, deeply buried in a template. Click this and a long list of messages opens.
It is the best to use clang, because the error messages are more helpful than those given by MSVC:
The last messages read:

Quote:
C:\upp\MyApps\test\test.cpp (53): note: in instantiation of member function 'std::__1::deque<SignalStatusMessage, std::__1::allocator<SignalStatusMessage>>::push_back' requested here
 (): deqSigStatus.push_back((sigStatus));
C:\upp\MyApps\test\test.cpp (24): note: copy constructor of 'SignalStatusMessage' is implicitly deleted because field 'status' has a deleted copy constructor
 (): Vector<SignalStatus> status;
C:\upp/uppsrc\Core/Vcont.h (288): note: copy constructor is implicitly deleted because 'Vector<SignalStatus>' has a user-declared move constructor
 (): Vector(Vector&& v)   { Pick(pick(v)); }
 (): 1 error generated.

Look in this list for notes about your own code. These indicate where the error could be caused, but it could only be detected later, when the template was evaluated.
If you know this, it becomes easier.

About the STL: I am not sure it is a design, it is evolutionary grown.
I have watched a lot of Bjarne Stroustrups lectures on youtube and the most important thing for him is progress, but equally important preserving backward compatibility and maintenance of some 20 years old very large code bases.
He discusses also the problem of unnecessary copies and how to avoid and which language and

STL features are in the pipeline in the hope to improve this.
I think "concepts" added to templates will in future help to detect and display such errors early.

Subject: Re: error: call to implicitly-deleted copy constructor
Posted by Novo on Tue, 23 Feb 2021 19:18:35 GMT
View Forum Message <> Reply to Message

NTL, which is now called Core, was created by Mirek ~20 years ago. He tried to promote it, but it was ignored.
10 years later we've got C++11, which is, basically, based on Mirek's ideas.
And yes, STL kept its disadvantages. It was original design flaw.
Fortunately, you do not have to use STL with C++. This is why we all are using U++

Subject: Re: error: call to implicitly-deleted copy constructor
Posted by mirek on Wed, 03 Mar 2021 09:04:58 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 23 February 2021 20:18NTL, which is now called Core, was created by Mirek ~20 years ago. He tried to promote it, but it was ignored.
10 years later we've got C++11, which is, basically, based on Mirek's ideas.

I would rather say "similar ideas".

Mirek
