Subject: Re: I request the implementation of callback5
Posted by Oblivion on Wed, 16 May 2018 07:10:38 GMT
View Forum Message <> Reply to Message

Quote:
it does not seem to be work..

Possibly a capture issue. I can't really say what's wrong until you provide the error messages. :)

```
threads[free_index].Run ( [&,a,bot1,result_mode, stat_group_id]
```

Not a good practice,really. Try to avoid using default capture by reference where possible. As it captures every variable in scope (This can give you headaches when your code gets complex.).

And as I said before, If you are capturing by reference, you need to take into account the lifetime of the captured object to prevent dangling references.

Capture by reference explicitly only the variables you need. And use the capture by value mechanism for others:

```
threads[free_index].Run ( [=,&a,&bot1,&result_mode, &stat_group_id]
```

By the way, using lambda's value copying is that thread safe?

Capture by value, copies (or moves where possible or explicitly stated) the parameters for each instance.
In general it is no different than ordinary copy operations, so yes.

(There are some corner cases though. I suggest reading Scott Meyers' Effective Modern C++ for information on the possible but rare complications.)

And again I suggest using CoWork if possible, as you seem to be dealing with worker threads.
A dumb, and simple example:

```
int n = 0;

CoWork co;

    for(int i = 0; i < 6; i++)
      co & [=, &n]{
            // Thread Code.
```

```
                CoWork::FinLock();
                n++;
        };
     co.Finish(); // Waits until all workers have finished their jobs. (There is also a non-blocking
version: CoWork::IsFinished())
```

Best regards,
Oblivion