Subject: Re: Question about PostCallback from Child Thread
Posted by mirek on Thu, 12 Jul 2007 19:54:49 GMT
View Forum Message <> Reply to Message

kfeng wrote on Thu, 12 July 2007 00:11OK, suppose I have a C struct with a bunch of pointers to the heap:

```
struct
{
  int    *intP;
  double *doubleP;
  char   **strP;
...
}
```

It's filled out by the child process and I pass a pointer to an instance to PostCallback(). Will PostCallback() be smart enough to lock all the pointed-to members? If not, is there a way to make the child thread block and wait until the parent is done reading?

The problem is **strP - I need this to run fast so I don't want to be looping through the members locking each one by hand - may be simpler to just get the child to wait for the parent to finish. Is there a way I can do this to a child thread?

Now I am a little bit confused.... First of all, "pointers to the heap" sounds like a bad practice if you are following U++ path of things.... Anyway, surely can happen.

Then the question is what "locking each one by hand" is supposed to mean.

In MT, what you have to lock (using mutex) is data that at specific moment can be accessible by more than single thread. Is this the case? Note that if you e.g. create heap data than are only passed using PostCallback and are not references anywhere else, you do not need to lock.

Now another question is why do you want to lock one by one? You can have single mutex protecting the whole array.... and lock just before the loop, unlock after.

Mirek