Subject: How does widget/variable destruction in U++ work?
Posted by crydev on Wed, 12 Dec 2012 09:53:28 GMT
View Forum Message <> Reply to Message

I am working on a program to manage movies on my computer. I have already built a lot, and I am testing all the time, also on memory usage. Now a question came to me. I use task manager to find out which actions take up memory, but I notice that when I execute a block as following:

```
void MovieManager::Options()
{
 SettingsDialog().Execute();
}
```

Which creates an instance of my settings dialog form, the memory usage increases by 600 KB. This is not weird, since I have a lot of controls, loading code and images inside there. However, when I close the dialog again by using:

```
Close();
```

The 600 KB memory that has been eaten is not being freed. Logically using C++ this way, the SettingsDialog variable that I created inside the function is destroyed as soon as the function finishes, and the used memory is freed.

Now I know that task manager is not the greatest tool to monitor memory usage as it is not quite as accurate as it should be. But this situation certainly brought up the question to me as it were: How does variable destruction and memory releasing work in U++?

Subject: Re: How does widget/variable destruction in U++ work?
Posted by dolik.rce on Wed, 12 Dec 2012 10:37:54 GMT
View Forum Message <> Reply to Message

crydev wrote on Wed, 12 December 2012 10:53The 600 KB memory that has been eaten is not being freed. Logically using C++ this way, the SettingsDialog variable that I created inside the function is destroyed as soon as the function finishes, and the used memory is freed.

Now I know that task manager is not the greatest tool to monitor memory usage as it is not quite as accurate as it should be. But this situation certainly brought up the question to me as it were: How does variable destruction and memory releasing work in U++?
Hi crydev,

U++ uses its own allocator, which works little bit different It allocates memory in chunks and then gives it to the variables that requested it (when new is called). When the memory is returned (delete is called) it keeps the allocated memory to allow future allocation to work faster. I'm not sure if it is describe in detail somewhere, but try searching the site and forum. Try repeating the action that allocated the memory again, the memory usage should not increase second time.

Also, IIRC the OS itself usually allocates and deallocates memory for application in larger chunks,

to lower the associated overhead.

To see detailed info about memory, you can use MemoryUsedKb() and some other U++ memory related functions.

Best regards,
Honza

Subject: Re: How does widget/variable destruction in U++ work?
Posted by dolik.rce on Wed, 12 Dec 2012 10:43:20 GMT
View Forum Message <> Reply to Message

PS: Also, in GUI apps you can call MemoryProfileInfo() to get a dialog with detailed memory info

---

Subject: Re: How does widget/variable destruction in U++ work?
Posted by crydev on Wed, 12 Dec 2012 10:53:47 GMT
View Forum Message <> Reply to Message

Quote:U++ uses its own allocator, which works little bit different It allocates memory in chunks and then gives it to the variables that requested it (when new is called). When the memory is returned (delete is called) it keeps the allocated memory to allow future allocation to work faster. I'm not sure if it is describe in detail somewhere, but try searching the site and forum. Try repeating the action that allocated the memory again, the memory usage should not increase second time.

Indeed, I also noticed that if I repeat opening the dialog the memory will not increase again. That clarifies the memory allocation system to me.

I will test the profile and memory functions for my application to find out how it works. Thanks for your reponse Honza, it is clear to me now

Crydev