Subject: Re: CTRL + C = 659 Heap leaks
Posted by mirek on Thu, 11 Jul 2019 22:15:22 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 11 July 2019 16:42mirek wrote on Thu, 11 July 2019 04:22
So, this is console application in POSIX and you are pressing Ctrl+C to terminate it. AFAIK, Ctrl+C just calls exit, which is what is causing the leaks, as 'mybot' never gets destroyed.

Ctrl+C sends SIGINT, which causes "soft" termination of an app (unlike SIGKILL). It does all cleanup job for the process including stack unwinding (at least in x64 Linux).

Proof:
Memory leak report is a result of a call to UPP::MemoryDumpLeaks(), and it is called by MemDiagCls::~MemDiagCls()

```
{
 if(--sMemDiagInitCount == 0)
  UPP::MemoryDumpLeaks();
}
```

So, stack gets unwinded, but in case of Xemuth memory doesn't get deallocated.
This is a bug with Xemuth's code. :blush:

I believe you are wrong. I believe Ctrl+C is equivalent of calling exit(). In that case, global variables get destructed, but local variables do not. This is also quite clear for the implementation of signals - they are asynchronouse and thus do not have to use the same stack so no stack unwinding is possible.

BTW, destructing global variables is the mechanism used to activate the leak checker :)

Mirek