Subject: Re: CTRL + C = 659 Heap leaks
Posted by mirek on Wed, 17 Jul 2019 18:25:13 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 17 July 2019 18:44mirek wrote on Tue, 16 July 2019 03:38
Maybe we are nor speaking about the same thing...

What I mean is

```
struct MemDiagCls {
 MemDiagCls();
 ~MemDiagCls();

 static void ShowReport() { show = true; }
 static bool IsShowReport() { return show; }

private:
 static bool show;
};
bool MemDiagCls::show = false;

MemDiagCls::~MemDiagCls()
{
 if(--sMemDiagInitCount == 0)
  if (IsShowReport())
   UPP::MemoryDumpLeaks();
}

struct LeakRepGuard {
 ~LeakRepGuard();
};

LeakRepGuard::~LeakRepGuard() {
 MemDiagCls::ShowReport();
}

#define CONSOLE_APP_MAIN \
void ConsoleMainFn_(); \
 \
int main(int argc, char *argv[]) { \
 UPP::LeakRepGuard __; \
 UPP::AppInit__(argc, (const char **)argv); \
 UPP::AppExecute__(ConsoleMainFn_); \
 UPP::AppExit__(); \
 return UPP::GetExitCode(); \
} \
 \
void ConsoleMainFn_()
```

A call to UPP::MemoryDumpLeaks() will be disabled if there was no stack unwinding. In all other cases it will work as before.
Leak report makes no sense when destructors of objects on stack were not called.
IMHO, this should be thread-safe because there is only one main() function ...

Ah, I see, in other words: "or make it disable leak checks completely in Win32 in Ctrl+C exception handler." (your method is more generic but the effect is the same).
