Subject: exit() -> heap leaks detected.
Posted by captainc on Fri, 05 Sep 2008 20:23:52 GMT
View Forum Message <> Reply to Message

Why does exit() cause a heap leak when Vector class has items?

Example:
```
CONSOLE_APP_MAIN
{
 Vector<String> v1;
 v1.Add("hello");
 exit(1);

}
```

Subject: Re: exit() -> heap leaks detected.
Posted by mr_ped on Fri, 05 Sep 2008 21:00:01 GMT
View Forum Message <> Reply to Message

Maybe "exit" does not call some destructors? Try to exit cleanly.
To set the exit value with clean finish you can use: (to see if it helps)

```
  UPP::SetExitCode( 1 );
```

BTW, what version of UPP and what compiler?
The leak detection should work ok just with the MSC. ( )

Subject: Re: exit() -> heap leaks detected.
Posted by captainc on Fri, 05 Sep 2008 21:15:37 GMT
View Forum Message <> Reply to Message

SetExitCode() does not perform the exit operation also.

My compiler is Linux GCC 4.1.3, but it also happens with MSC8.
Upp SVN.324

I first encountered it when trying to use another C++ library that called exit(). I can control the exit path on my application, but I don't want to have to modify other libs. I partially wrote the command line processor package for this reason.

Subject: Re: exit() -> heap leaks detected.
Posted by mr_ped on Fri, 05 Sep 2008 21:44:56 GMT
View Forum Message <> Reply to Message

captainc wrote on Fri, 05 September 2008 23:15SetExitCode() does not perform the exit operation also.

I was sort of trying to write that, but reading my post again ... and it's far from obvious.

Well, I was unable to make leak detection work in 2008.1 with GCC (MINGW too). I mean, I forced it to run and collect data, but the init of U++ itself did leak always. Maybe the SVN version is fixed a bit in this manner?
I can't follow the SVN, I need very stable code base for my purposes, and I don't have spare time to try it in other directory right now. :/

Are you sure the Vector is leaking? When you remove the Vector stuff, does it leak or not? (and you can use the memory breakpoint feature to break at the leaking alloc ... although debugging with GCC is total pain, I rather add some LOG here/there to avoid debugging with U++, it's so bad. :/

Subject: Re: exit() -> heap leaks detected.
Posted by mirek on Fri, 05 Sep 2008 22:07:36 GMT
View Forum Message <> Reply to Message

I think it is quite obvious:

How do you expect the Vector destructor (of *local* variable) being called? The control never gets to the end of block....

The moral of the story is: never call "exit" in C++ and expect clean exit...

Mirek

Subject: Re: exit() -> heap leaks detected.
Posted by captainc on Fri, 05 Sep 2008 22:39:53 GMT
View Forum Message <> Reply to Message

From definition of exit function:
Quote:Terminates the process normally, performing the regular cleanup for terminating processes.

First, all functions registered by calls to atexit are executed in the reverse order of their registration. Then, all streams are closed and the temporary files deleted, and finally the control is returned to the host environment.
This is why I have been confused. Is the definition not consistent with the actions? I guess I did not research it enough... neither did a few library authors!

Subject: Re: exit() -> heap leaks detected.
Posted by captainc on Fri, 05 Sep 2008 22:43:57 GMT
View Forum Message <> Reply to Message

Seems with Upp, the right course of action would be to:
SetExitCode(1);
return;
instead of using exit(1);

As we are really in function ConsoleMainFn_(), which is called from main().
hmm, but this doesn't solve library issue.

Subject: Re: exit() -> heap leaks detected.
Posted by mirek on Sat, 06 Sep 2008 06:39:26 GMT
View Forum Message <> Reply to Message

captainc wrote on Fri, 05 September 2008 18:39From definition of exit function:
Quote:Terminates the process normally, performing the regular cleanup for terminating processes.

First, all functions registered by calls to atexit are executed in the reverse order of their registration. Then, all streams are closed and the temporary files deleted, and finally the control is returned to the host environment.
This is why I have been confused. Is the definition not consistent with the actions? I guess I did not research it enough... neither did a few library authors!

Well, global destructors are registred via atexit (usually, it is). Anyway, "exit" never returns, means nothing can be called at the end of block.

OTOH, as system usually performs necessarry cleanup (closing file handles etc...), I can see that some C++ library calls exit, not caring about leaks. But that is not really correct in my book.

Mirek

Subject: Re: exit() -> heap leaks detected.
Posted by mirek on Sat, 06 Sep 2008 06:41:44 GMT
View Forum Message <> Reply to Message

captainc wrote on Fri, 05 September 2008 18:43Seems with Upp, the right course of action would be to:
SetExitCode(1);
return;
instead of using exit(1);

As we are really in function ConsoleMainFn_(), which is called from main().
hmm, but this doesn't solve library issue.

IMO, you cannot solve the problem without altering the library code (maybe you should complain to the author

Possible fix is to use exception handling - define something like "struct Exit", throw it instead of "exit" and catch in main, then SetExitCde(1), return.

Mirek

Subject: Re: exit() -> heap leaks detected.
Posted by captainc on Sat, 06 Sep 2008 11:25:02 GMT
View Forum Message <> Reply to Message

Quote:IMO, you cannot solve the problem without altering the library code (maybe you should complain to the author
Yeah, I really don't like the fact that they call exit instead of throwing an exception. I would rather alter it or find some other way.
