Subject: Heap errors behavior is dependent on target machine.
Posted by jfranks on Sat, 28 Nov 2015 13:03:33 GMT
View Forum Message <> Reply to Message

Brief Description: Memory heap errors don't happen for development
machine 'A'. However, they do occur for the same
executable image on the embedded target machine
'B'.

Full Description of the issue:

Description of development machine -- i.e., machine 'A'

The development machine is a Dell desktop computer that has an Intel CORE i5, and a host operating system Windows 7. A virtual machine was installed using Oracle VM Virtual Box to run a guest operating system Linux Mint 17.2. Our application development was done on this virtual machine as a Linux based application using U++ nightly snapshot upp-x11-src-9200.

Description of the embedded target machine -- i.e., machine 'B'

This is proprietary custom hardware that has a touchscreen, custom keypad entry device, a commercial power supply, commercial single board computer, and a hard drive. The operating system is the same as that used on the development machine. The CPU is compatible to run the executable image produced on the development machine.

Description of the problem we are having with U++ memory diagnostic:

1. We developed and debugged our graphical U++ application on machine 'A'. All memory heap errors were located and corrected. The executable image developed on this machine is compatible for running on machine 'B'.

2. We installed the debug version of our executable image on machine 'B'. Everything runs great except when we exit our application. The behavior is different on machine 'B' in that there are memory heap errors, followed by a segfault on X11.

```
Heap leaks detected!
Segmentation fault
```

3. We enabled machine 'B' to have development capability and installed U++ IDE based on upp-x11-src-9200. We did a code checkout into this machine from our SVN server. We compiled the code. Then we ran the debug executable built on this machine. The result was the same as item 2 above.

The debug executable image built on machine 'B' was copied to machine 'A' and it exhibited a different behavior -- it worked correctly on exit from the application, i.e., there were no memory heap errors, nor segfault. That is odd.

Next, we decided to start debugging on machine 'B' in earnest. We modified our application code and inserted MemoryIgnoreLeaksBegin(); and MemoryIgnoreLeaksEnd(); so as to exclude all of our application code from leak detection. The result was the same as in item 2 above.

We more aggressively applied the U++ memory ignore function by reworking the GUI_APP_MAIN macro and explicitly replaced it with the following.

```
//GUI_APP_MAIN {
void GuiMainFn_();
int main(int argc, const char **argv, const char **envptr)
{
MemoryIgnoreLeaksBegin();
 UPP::AppInit__(argc, argv, envptr);
 UPP::Ctrl::InitX11(NULL);
 UPP::AppExecute__(GuiMainFn_);
 UPP::Ctrl::ExitX11();
 UPP::AppExit__();
MemoryIgnoreLeaksEnd();
 return UPP::GetExitCode();
}

void GuiMainFn_()
{
 ... our application code starts here ....
}
```

The results on machine 'B' did not change -- still a memory heap issue on exit and a segfault. A large log file was produced with many memory breakpoints.

Next, we compiled a release version of the application on machine 'B' without any debug flags. Everything works great because the U++ memory diagnostics are disabled. As we run the release version, there is nothing that indicates a problem at any time, even when we exit.

The log file generated from running the debug was over 2400 items. I've attached a snapshot of the call-stack while in the debugger for the lowest numbered memory break-point #1.

We are having a difficult time sorting this out and are asking

for help or ideas of where we go from here.

-- Jeff

File Attachments

1) call-stack.jpg, downloaded 650 times

Thread	Frame	Function	Arguments	Loc
4	0	__vsnprintf_chk	()	
2	1	__snprintf_chk	()	
1	2	XauFileName	()	
	3	XauGetBestAuthByAddr	()	
	4	??	()	
	5	??	()	
	6	xcb_connect_to_display_with_auth_info	()	
	7	xcb_connect	()	
	8	_XConnectXCB	()	
	9	XOpenDisplay	()	
	10	Upp::sPanicMessageBox	(title = 0x9126ce6 "Fatal error", text = 0x91247f2 "Heap leaks detected!")	/hor
	11	Upp::PanicMessageBox	(title = 0x9126ce6 "Fatal error", text = 0x91247f2 "Heap leaks detected!")	/hor
	12	Upp::Panic	(msg = 0x91247f2 "Heap leaks detected!")	/hor
	13	Upp::MemoryDumpLeaks	()	/hor
	14	MemDiagCls::~MemDiagCls	(this = 0x93556e4 <sMemDiagHelper__upp__>)	/hor
	15	??	()	
	16	exit	()	
	17	__libc_start_main	()	
	18	_start	()	