Subject: Re: Heap errors behavior is dependent on target machine.
Posted by jfranks on Tue, 01 Dec 2015 04:58:39 GMT
View Forum Message <> Reply to Message

Great work!! Thank you so much for your help.

1. Regarding your instructions -- Mon, 30 November 2015 10:03 . . .

The results of that test is as follows:

$ ./p101-dbg
Module initialized
Now checking for leaks
Heap leaks detected!
Segmentation fault

The destructor for the shared serial library was not yet called
when heap leaks were being evaluated.

Regarding details that you requested . . .

Q1. What is that "compatible" CPU?

A1. Celeron J1900 on an IMB-151 single board computer.
Reference: http://www.asrock.com/ipc/overview.asp?Model=IMB-151

Also, on machine 'B', "uname -a" provides the following:

Linux administrator-desktop 3.19.0-28-generic #30~14.04.1-Ubuntu \
SMP Tue Sep 1 09:32:55 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux

Q2. Is the system updated to current version and is it exactly the same?

A2. Machines 'A' and 'B' are using Linux Mint 17.2 distros. There
is one serial port sharable library (32-bit) that is not part
of the distro that was installed on both machines.

Q3. Are there any peripherals using serial communications that are
not on Dell?

A3. Yes. The installation of our application on the embedded
target machine 'B' uses serial port #1 for logging, and serial
port #2 for Modbus communications. So, two serial ports exist
in hardware for machine 'B'. However, the application can run
without these serial ports, since logging and Modbus are
optional items.

The development Machine 'A' is just a common desktop computer. Serial ports have not been made available to the guest virtual machine, which is where we are developing and testing. When the app runs and discovers no serial logging port and/or no Modbus port, these functions are just turned off.

Also, the hardware for the embedded system machine 'B' has additional devices and Linux drivers to handle a custom keypad HID device, and a touchscreen.

Q4. What is that shared library?

A4. libserial-0.5.2 ... attached to this response. We have compiled this as a 32-bit library so that it matches our 32-bit application for both machine 'A' and 'B'.
libserial-0.5.2/src/SerialPort.cpp is where those stdc++ strings are located the U++ heap diag was flagging as a memory leak.

---

2. Regarding your instructions on Mon, 30 November 2015 10:05 . . .

The results of testing with heap dump disabled:

```
$ ./p101-dbg
Module initialized
Now checking for leaks
Module deinitialized
```

This confirms that the destructor for the serial port shared library is called after heap leaks has been evaluated.

static const MemDiagCls sMemDiagHelper __attribute__ ((init_priority (0)));
caused a compilation error (out-of-range).
I changed the 0 to a 1, and the compiler gave a warning.

The results of testing with init_priority (1) was the same as previous:

```
$ ./p101-dbg
Module initialized
Now checking for leaks
Module deinitialized
```

---

3. Regarding your instructions on Mon, 30 November 2015 10:45 . . .

The modified Core/heapdbg.cpp file was downloaded and replaced
the previous one in my copy of nightly build 9246.

This file was obtained from

https://github.com/ultimatepp/mirror/blob/1ce7608b2fb7571902
917401d4215fb76f03eafd/uppsrc/Core/heapdbg.cpp

Results: There is an improvement !! Heap errors related to stc++
strings disappeared from the log file. However, we are
still experiencing heap errors related to something else.

The pattern of heap leaks with sizes of 812 and 828 are
still there. The number of these varies.

Conclusion:

- memory heap leaks related to stdc++ strings
are resolved with the modifications that you made
to Core/heapdbg.cpp

- memory heap leaks are still being detected
from some other source.

BTW: The above result was double checked.

- Core/heapdbg.cpp was reverted to the original. Retesting showed
the last 7 items in the log file related to stdc++ strings
reappeared. This is what is expected for this test scenario.

- Core/heapdbg.cpp was again replaced with the new modified
version. Retesting showed for this case that heap errors
previously related to stc++ strings disappeared.
This is good and is an improvement.

Conclusion:

The memory heap diagnostic is improved so that stdc++ strings
in shared library are ignored (that is good).

There is another source of heap errors. Let's say that these
are all from a similar source because they always come in pairs and
the total number for each size (812 and 828) are always the same.

Is there a strategy that can be applied to be able to identify something about where these originate?

Summary:

One down, one to go.

-- Jeff

File Attachments

1) SerialPortLib.tar.gz, downloaded 332 times