Quick way to find memory leaks in C/C++programs#

If your application is leaking memory the best way to find out where it is leaking is to use Memory Analysis tool.

For first iteration you only need

- you binary located on the target and ready to run
- TCP/IP connection to the target and qconn agent running on it (see below for workaround if you don't have a direct connection)
- librcheck.so library installed on the the target (latest version can be download here)

Finding leaks by attaching to application#

This would describe how to create launch configuration to find memory leaks in a binary that was not build in IDE and cannot be launched from IDE.

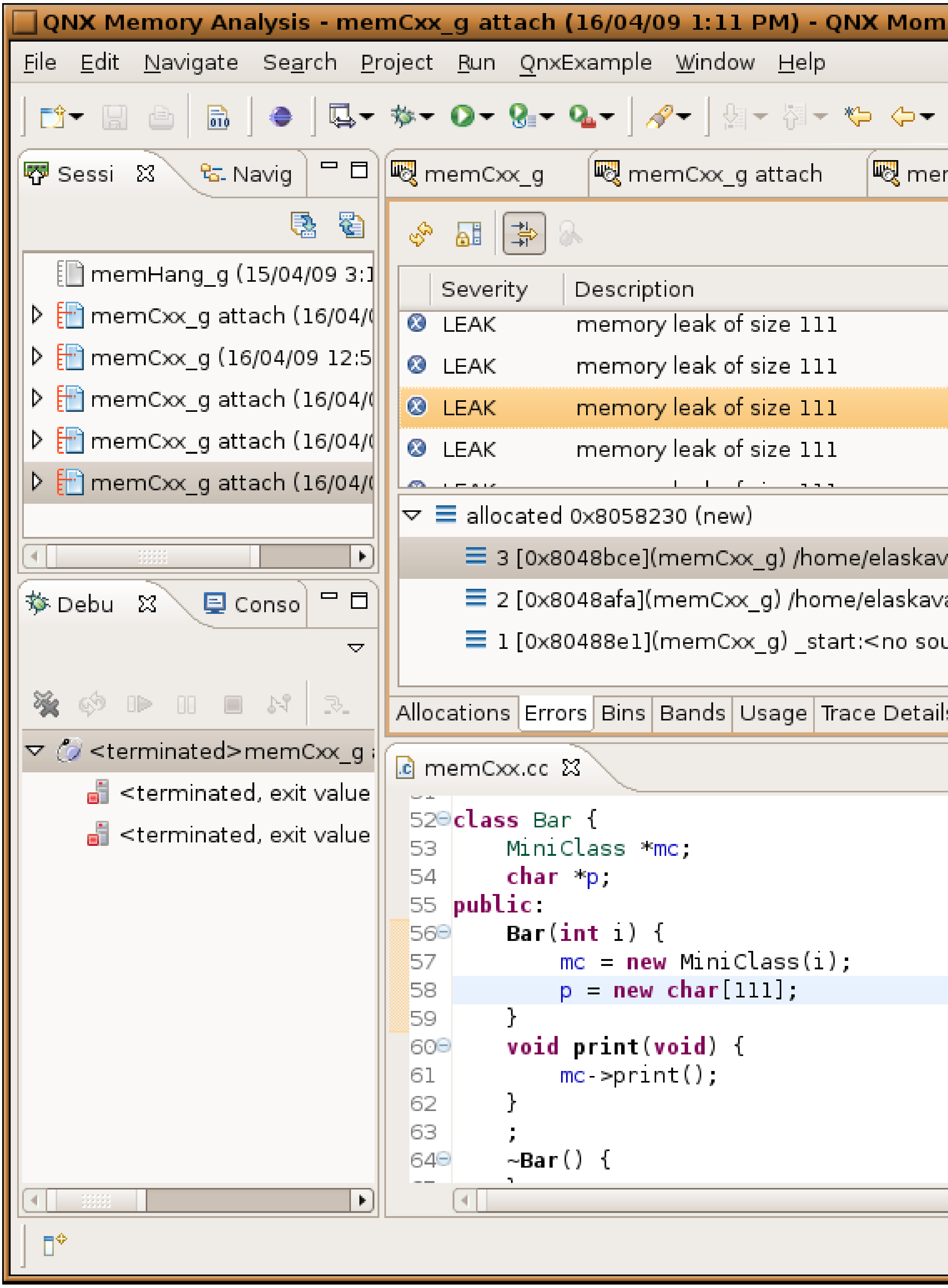
- In IDE create a C Makefile project (New->Projects...->C Project->Makefile->Empty Project (QNX Toolchain))
- Open Target Navigator and create a New target with IP address of the target you going to be running program
- Copy you binary from target to a host and place it in the created project (you can use Target File System Navigator for that)
- Right click on binary and select Profile As...->C/C++ QNX Application Dialog...
- On the Main tab select your target
- Switch to Tools tab and Add Memory Analysis tooling, Enable switch to this tool's perspective (checkbox)
- Click Apply and Close (don't click on Profile button - nothing is running yet)

Now go back to target terminal:

Modify you launch script to include extra environment variables (and remove trace file, if exists): rm /tmp/traces.rmat; LD_PRELOAD=librcheck.so MALLOC_CTHREAD=1 MALLOC_TRACE=1 MALLOC_FILE=/tmp/traces.rmat ./your_app <arg>

and run it, switch back to IDE

- In IDE right click on binary and select Profile As...->C/C++ QNX Application Dialog...
- Configuration is already prepared so just click "Profile" button
- Pick the process name and click Ok
- IDE should open Memory Analysis perspective
- You should see new session created in Sessions view. Double click on it.
- Let your app run, maybe execute a test that makes it leak memory
- Now switch to Settings tab and press get Leaks button
- Switch to Errors tab and you should see Leak errors there if you have any. Click on error line to see allocation backtrace. This object has no references and has not being freed.


QNX Memory Analysis - memCxx_g attach (16/04/09 1:11 PM) - QNX Mom
File Edit Navigate Search Project Run QnxExample Window Help
Sessi
Navig
memCxx_g
memCxx_g attach
mer
Debu
Conso
Allocations
Errors
Bins
Bands
Usage
Trace Details
memCxx.cc

memHang_g (15/04/09 3:1
memCxx_g attach (16/04/(
memCxx_g (16/04/09 12:5
memCxx_g attach (16/04/(
memCxx_g attach (16/04/(
memCxx_g attach (16/04/(

Severity	Description
LEAK	memory leak of size 111
LEAK	memory leak of size 111
LEAK	memory leak of size 111
LEAK	memory leak of size 111

```
allocated 0x8058230 (new)
    3 [0x8048bce](memCxx_g) /home/elaskav
    2 [0x8048afa](memCxx_g) /home/elaskava
    1 [0x80488e1](memCxx_g) _start:<no sou
```

```
<terminated>memCxx_g
    <terminated, exit value
    <terminated, exit value
```

```cpp
class Bar {
    MiniClass *mc;
    char *p;
public:
    Bar(int i) {
        mc = new MiniClass(i);
        p = new char[111];
    }
    void print(void) {
        mc->print();
    }
    ;
    ~Bar() {
```

Finding leaks by running from IDE#

- If you don't use IDE to build your binary create simple makefile project and drop your binary in there
- Select it, right click and select Run as QNX Application Dialog...
- Select target, Environment, Arguments
- Switch to Tools tab and Add Memory Analysis tooling, Enable switch to this tool's perspective
- When app started, double click on "session" in Sessions view of Memory Analysis perspective, it would open an graphical editor
- Let your app run, maybe execute a test that makes it leak memory
- Now switch to Settings tab and press get Leaks button
- Switch to Errors tab and you should see Leaks errors there if you have any. Click on error line to see allocation backtrace. This object has no references and has not being freed.

What to do if backtrace is not there or has not enough data#

If you see a leak but it does not point to source code or there is no backtrace:

- You need debug version of the binary. You can run non-debug on target but host site has to have debug symbols (must be same build otherwise)
- If you see incomplete trace it can be due to compiler optimization, compile with -O0
- This code can belong to a library, if it is your library you can add it to IDE search path, buy adding path in Shared Library tab of Tools tab (Add tab using Add/Delete Tool... button) of launch configuration. Library has to have debug symbols too for source to show up.
- To capture backtrace memory tracing has to be on (MALLOC_TRACE=1 from command line).
- By default backtrace is depth of 5, if you need more you can change IDE settings in Tools tab or set MALLOC_TRACEBTDEPTH=10 from command line.

What to do if you don't have TCP/IP connection#

- You can find leaks postmortem by transferring a log file to IDE
- Run your application as `rm /tmp/traces.rmat; LD_PRELOAD=librcheck.so MALLOC_CTHREAD=1 MALLOC_TRACE=1 MALLOC_FILE=/tmp/traces.rmat MALLOC_DUMP_LEAKS=1 ./your_app <arg>`
- This would dump leaks then app exists normally. If you kill the app it does not "exit normally".
- If you want to initiate leak detection before exit you can use "control thread" interface
- Run `echo trace_dump_unref /tmp/traces.rmat > /dev/rcheck/ctl/1982503`, where 1982503 should be replaced with your process ID (usually it would be just one there so you can use /dev/rcheck/ctl/*)
- Transfer /tmp/traces.rmat file on the host
- From IDE switch to Memory Analysis perspective and in Session view select an Import command. Select create a new session to import data into. Select a file to import, debug binary and shared library path if you have any of your libraries involved. Click Finish.
- Double click on new session and switch to Errors tab to see memory leaks.

If you need more details please consult User Documentation.