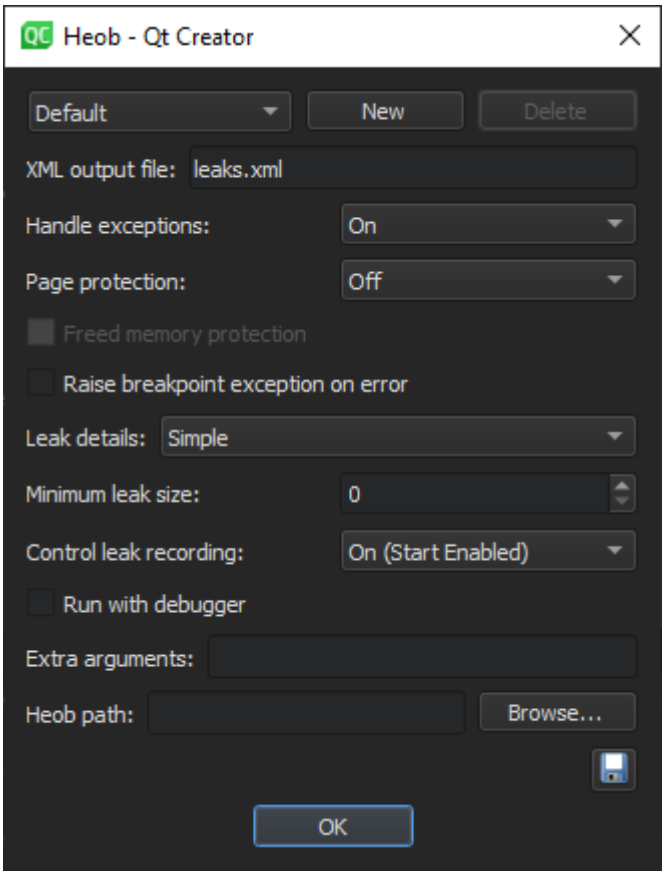
使用 Heob 检测内存泄漏

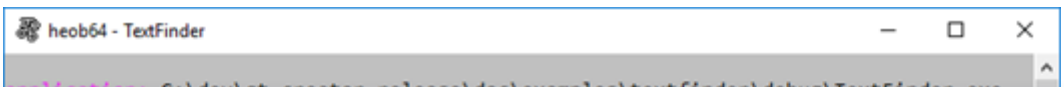
Qt Creator 集成了 Heob 堆观察器，用于检测缓冲区溢出和内存泄漏。您必须下载并安装Heob才能从Qt创建器运行它。

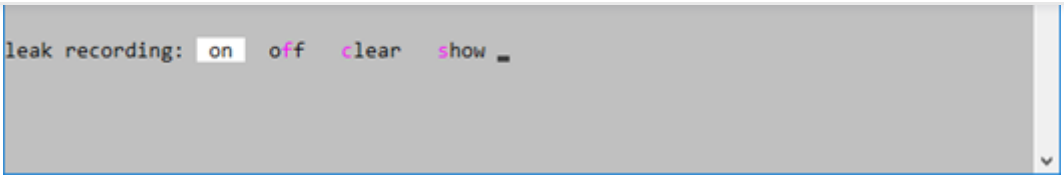


要在当前打开的项目上运行 Heob，请执行以下操作：

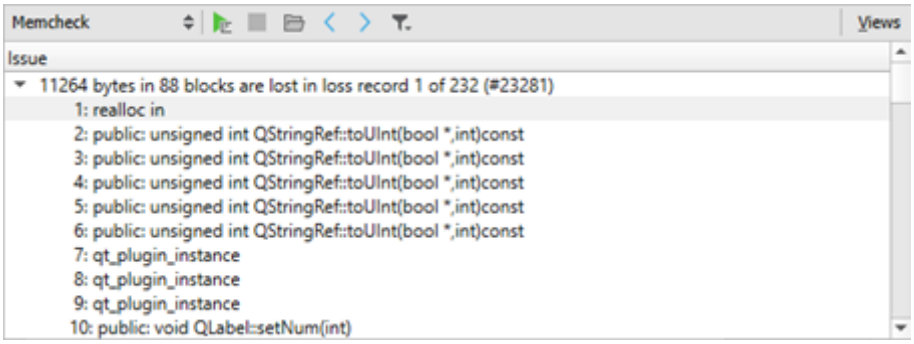
1. 选择"**分析**> **Heob**"。
2. 选择要使用的 Heob 设置配置文件，或选择"**新建**"以创建新的配置文件。
3. 在"**Heob 路径**"字段中，输入 Heob 可执行文件的路径。
4. 指定用于运行检查的其他设置。有关可用选项的详细信息，请参阅指定 Heob 设置。
5. 选择"**确定**"以运行 Heob。

Qt 创建器运行应用程序，然后在控制台窗口中运行 Heob。





Heob 在缓冲区溢出时引发访问冲突，并提供违规指令和缓冲区分配的堆栈跟踪。当 Heob 正常退出时，将显示结果。



指定赫布设置

若要指定 Heob 的设置，请选择“**分析**> **Heob**”。

在“**额外参数**”字段中，输入用于运行 Heob 的其他参数。若要在 Heob 控制台中列出可用参数，请在此字段中输入，然后按 **Enter**。 -H

例如，使用该选项在 HTML 文件中记录泄漏数据。与此选项一起，您可以使用该选项在文件中直观地对泄漏进行分组，并使用该选项在文件中记录最多 1024 个字节的泄漏内容。例如`-oleaks.html-g2-L1024-oleaks.html -g2 -L1024`

若要保存设置配置文件，请选择“**保存**”。

若要删除自定义设置配置文件，请选择该配置文件，然后选择“**删除**”。

以下各节更详细地介绍了可用选项。

记录结果

检查结果显示在“**Memcheck**”视图中，并记录在文件中。您可以在 **XML 输出**文件字段中指定文件名。Heob 在项目目录中创建该文件。

You can use the process identifier (PID) as a variable in the file name. For example, . This injects Heob into the child processes, as well.`leaks-%p.xml`

Other variables you can use are for the parent PID and for the application name.%P%n

If you use variables, Qt Creator cannot open the file automatically, but you can open it from the project directory.

Handling Exceptions

In the **Handle exceptions** list, select **Off** to use the standard exception handler and have the debugger automatically attached if the application crashes. This works only if you register Qt Creator is as a post-mortem debugger by selecting **Edit** > **Preferences** > **Debugger** > **General** > **Use Qt Creator for post-mortem debugging**.

Select **On** to use the Heob exception handler that checks the reason and location of the crash and detects whether it occurred because of buffer overrun.

stack trace of the crash is displayed. Therefore, this option is mostly useful when using Heob on the console or running it for child processes, as well.

Raising Exceptions on Errors

Select the **Raise breakpoint exception on error** check box to display errors when the application runs.

If the option is disabled, errors such as *double free*, *free of invalid pointer*, and *not enough memory* just write all collected data into the results file and you will only see them at the end of the application run.

If the option is enabled, the application stops at the error location. This is mostly useful when used with the **Run with debugger** option, which runs Heob under the debugger.

Protecting Pages

In the **Page protection** list, select **Off** to use standard memory allocation functions and enable only memory leak detection.

Select **After** to place a protected page at the end of each allocated block and throw an exception if it is accessed. Select **Before** to place a protected page before each allocated block. These options consume memory and slow down the checks, and are therefore recommended only for 64-bit or short-running programs.

Select **Freed memory protection** to protect all allocation pages when freed, so that their address space can never be used again. This is useful for *use-after-free* and *double-free* detection. However, the available memory address space can run out fast for 32-bit programs.

Handling Leak Data

In the **Leak details** list, determine how to handle the collected leak data when the process exits. Selecting **None** means that no leak data is collected. If you activate leak type detection, Heob might need more time to collect the data when the process exits.

Select **Simple** to write all memory that was not freed into the results file.

Select **Detect Leak Types** to parse all static and global memory blocks for references to the leaks. The reachable blocks are marked *reachable* and recursively checked for other references. If references are found, the blocks are marked *indirectly reachable*. The remaining blocks are checked for references to each other and marked either *indirectly lost* or *jointly lost* (if the blocks reference each other). The blocks that have no references at all are marked *lost*. Select **Detect Leak Types (Show Reachable)** to also record the the reachable blocks in the results file.

Select **Fuzzy Detect Leak Types** to mark memory blocks *reachable* or *indirectly lost* if they contain references to any address. This option is useful when used with some custom allocators (such as in) that keep only an address somewhere inside the allocation block and do not refer directly to the start of an allocated block. Select **Detect Leak Types (Show Reachable)** to also record the reachable blocks in the results file.`av_malloc()``ffmpeg`

In the **Minimum leak size** list, select the size of leaks to detect in bytes.

In the **Control leak recording** list, select **Off** to record all leaks. You cannot change leak recording while it is running.

To start Heob without starting leak recording, select **On (Start Disabled)**. In the Heob console, turn recording **on** or **off**, **clear** all results, or select **show** to record all current leaks into the results file. Open the file to see its contents before the process exits.

To start leak recording when Heob starts and still have the option to control the recording, select **On (Start Enabled)**.

Qt DOCUMENTATION

Contact Us

Company

About Us
Investors
Newsroom
Careers
Office Locations

Licensing

Terms & Conditions
Open Source
FAQ

Support

Support Services
Professional Services
Partners
Training

For Customers

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

Feedback Sign In