数据断点

*此示例与2015年11月的Xbox One XDK或更高版本兼容*

# 描述

此样本展示如何创建对检测不同类型的内存访问有用的硬件数据断点。它们完全由处理器核心处理，这意味着它们不会影响执行速度。使用它们的主要缺点是处理器核心上只有 4 个可用插槽。

Windows 仅允许线程的本地断点。这意味着断点仅在相关线程正在执行时才处于活动状态。由于硬件断点绑定到线程，因此，其将在各个核心上执行时跟随线程。如果需要，可以使用相同的断点设置每个线程。

# 目录

## DataBreak.cpp/h

* 独立程序包，可设置和清除硬件数据断点。
* SetHardwareBreakPointForThread
  + 在给定相关插槽和地址的前提下，设置断点。
  + 已用的插槽将被覆盖。
* ClearHardwareBreakPointForThread
  + 清除指定插槽上的断点

# 实施说明

要设置硬件断点，需要调整调试寄存器的内容。主要问题是仅内核可访问这些寄存器。设置调试寄存器的技巧是调整线程上下文。这将导致调度程序在上下文切换期间将内容写入调试寄存器。

只有在线程被挂起时才能更改线程的上下文。因此，如果目标线程是当前线程，则需要另一个线程来执行操作。此样本将创建一个临时工作线程以执行上下文切换。

抛出的异常可以通过结构化异常处理 **(\_\_try**, **\_\_except**) 或通过未处理异常过滤器来捕获。但是，在与**MiniDumpWriteDump** 组合使用时，捕获异常记录会被忽略。但是，如果正在引起异常的线程上调用 **MiniDumpWriteDump**，则有问题的代码仍将在调用堆栈中，只是更进一步。在有问题线程的上下文中调用未处理异常过滤器。

## 重要提示：

此系统抛出的异常是单步异常，由 OS 专门处理。如果连接了调试器，则调试器将首先捕获它。默认情况下，Visual Studio 将忽略它，控制权将传递给游戏异常处理程序，除非您单步执行代码。默认情况下，KD 将在导致异常的代码行处中断。如未连接调试器，则 **EnableKernelDebugging** 定义该行为。如果其已启用，则 ERA 将冻结。控制台会等待连接调试器。如果 **EnableKernelDebugging** 未启用，则将调用游戏异常处理程序。

DataBreak.cpp 中的 **DataBreakThread** 函数记录调试寄存器所需的内容。

# 更新历史记录

2016年10月初次发布