Speaker Introduction

- T.Roy
  - Masters Degree in Computer Engineering
  - 20 years experience in system software development
  - 10 years international teaching experience
  - Specialization in Windows Driver Development and Debugging
  - Founder of CodeMachine
- CodeMachine Inc.
  - Consulting and Training Company
  - Based in Palo Alto, CA, USA
  - Custom Driver Development and Debugging Services
  - Corporate on-site training in Windows Internals, Networking, Device Drivers and Debugging
  - http://www.codemachine.com

CodeMachine Courses

- Internals Track
  - Windows User Mode Internals
  - Windows Kernel Mode Internals
- Debugging Track
  - Windows Basic Debugging
  - Windows User Mode Debugging
  - Windows Kernel Mode Debugging
- Development Track
  - Windows Network Drivers
  - Windows Kernel Software Drivers
  - Windows Kernel Filter Drivers
  - Windows Driver Model (WDM)
  - Windows Driver Framework (KMDF)

Why This Talk…

- The problem
  - Developer and Technical support folks have to deal with crashes and hangs day in & day out
  - In many cases ONE crash dump is all they have to root cause a problem
  - Often critical pieces of information that are required to nail down a problem is missing from that one crash dump

So what can the developers do to help the support folks do their job better and faster ?

- This talk covers some simple programming techniques
  - To improve diagnosability of your code
  - To help support folks get more out of the crash dumps
  - To enable them determine root cause of an issue from a single crash dump
    - So they don't have to ask the customer to reproduce the problem again to get them yet another crash dump

Key Takeaways…

- In-memory data logging
- Preventing overwrite of important information
- Making data easily locatable and identifiable
- Logging relevant data and presenting it properly
- Complementing the OS's data tracking
- Understanding OS support for run time data capture
- Capturing performance related data

Techniques discussed here clearly apply to kernel mode drivers but …

They can be easily adapted to user mode code as well

Agenda

- Memory Trace Buffers
- Freed Pool Memory
- Structure Tracking
- Information Presentation
- State Logging
- Lock Owners
- Run Time Stack Traces
- Timing Information

Memory Trace Buffers

- Crash Dumps offer a temporal snapshot of a system
  - Provides no historical information
  - Often historical events are critical to root causing issues
- Log run time information into memory trace buffers
  - Non-Paged buffers available in kernel and complete dumps
  - Use circular buffer with wrap around feature
    - Retains most recent events by replacing old ones
    - Good compromise between memory usage & history length
  - Avoid locking when logging events in memory
    - Costly due to IRQL changes
    - Use Interlocked operations instead
- Trace buffer information can be retrieved using ‘dt –a’
- Enable/Disable logging code using registry keys
- Kernel internally uses this type of logging
  - Example : In-Flight Recorder (IFR) Logs
  - Example : PnP State History inside Device Node (DEVNODE)

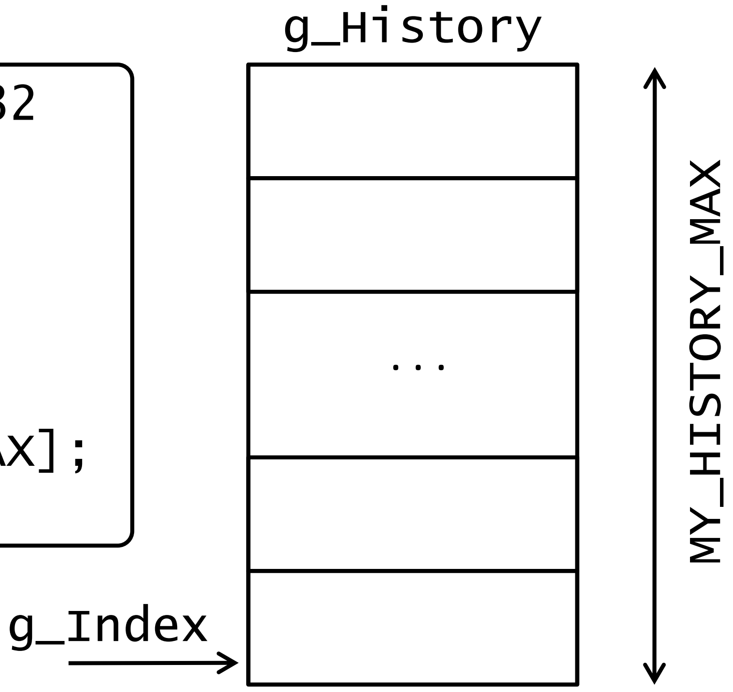
Implementation

- Data Structures

```
#define MY_HISTORY_MAX                    32

typedef struct _MY_HISTORY {
       PVOID Information;
} MY_HISTORY, *PMY_HISTORY;

MY_HISTORY g_History[MY_HISTORY_MAX];
ULONG g_Index = 0;
```



- Function

```
LoggingFunction( PVOID Information )
{
       ULONG Index = InterlockedIncrement (&g_Index);
       PMY_HISTORY History =
              &g_History[Index % MY_HISTORY_MAX];
       History->Information = Information;
}
```