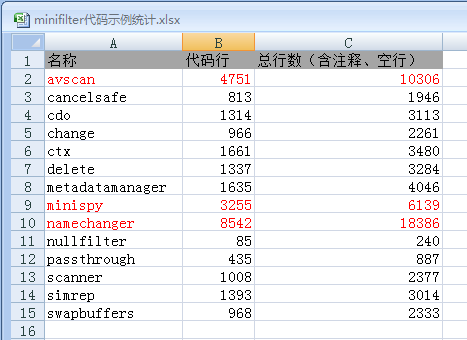
WDK minifilter 例子简要说明

# 一、代码统计



# 二、简要说明

* avscan：比ctx的例子还多了FLT\_TRANSACTION\_CONTEXT, FLT\_SECTION\_CONTEXT 的部分，也就是说多了事务和Section的处理，演示了各种context和锁的使用，使用FltRegisterForDataScan() 函数注册filter。
* cancelSafe：未看。
* cdo：未看。
* change：未看。
* ctx：演示了各种context的创建，获取和使用，包括 Instance Context，File Context，Stream Context，Stream Handel Context等。
* delete：未看。
* MetadataManager：未看。
* minispy： 可以指定监控和记录某个磁盘分区的I/O或内存事务活动的工具，默认显示的监控信息很少，要手动执行命令绑定并且显示log，可以选择把log信息保存到文件里。但说实话，记录的信息很难看懂。
* NameChanger：未看。
* nullFilter：这是一个空的miniFilter工程，没什么内容，适合初学者或演示用。
* passThrough：一个miniFilter的演示例子，虽然有代码，但是所有回调只是DEBUG输出一下函数信息就退出了，没什么内容，适合初学者或演示用。
* scanner：未看。
* simrep：未看。
* swapBuffers：演示如何在IRP\_MJ\_READ和IRP\_MJ\_WRITE中如何交换缓存，然后实现对新缓存的操作，操作完了再拷贝到旧的缓存里，并恢复。因为旧的缓存可能没有修改（写）的权限，所以使用这样的方法来实现对缓存的读写操作。Antinvader里的实现很接近该例子，但细节上有些小问题。

（注：标记为红色且我没看过的是可能需要着重看的例子，我已经看过且标记为红色的是比较重要的例子。我没看过的可以参考下面的说明。）

# 三、之前整理的说明

**cancelSafe minifilter sample**

描述：如果你要使用 cancel-safe 队列（什么意思？），就用 cancelsafe minifilter sample吧。

操作原理：当 cancelsafe minifilter 附属于一个卷时，将初始化一个 cancel-safe 队列。当这个 minifilter 销毁时，监控器读取通过 I/O 栈的操作。如果这个读操作已经在一个名叫 csqdemo.txt 文件上运行，这个操作将被排到 cancel-safe 队列，排队操作将在短暂的停顿（通过一个单独的运行在系统上下文的工作线程）之后完成。（去死啊，搞那么多that）

**minispy minifilter sample**

描述：本示例是一种监控和记录任何 I/O 以及重现系统内部事务活动的工具。这个例子类似大名鼎鼎的 filespy，不过它是用 minifilter 实现的。

操作原理：minispy 由用户模式和内核模式组成，内核模式组件注册（相当于各种利用过滤管理器进行 I/O 和事务操做）的回调函数，

这些回调函数帮助 minispy 记录任何系统的 I/O 和事务动作事件。当一个用户能够访问记录信息时，这个记录信息通过用户模式组件，

即可以输出到屏幕上也可以输入到磁盘上的 log 文件里要观察一个设备上的 I/O 活动，你必须明确地指定 minispy 和这个设备的联系，同样，

也可以要求 minispy 停止对一个特定设备记录日志。

**ctx minifilter sample**

描述：本例用来说明如何指派你的 minifilter 和实例、文件、流或流句柄的上下文。

操作原理：ctx minifilter 展示如何从和实例、文件、流或流句柄指派上下文或移除上下文，无论一个或多个对象被创建，ctx 都会指派一个上下文。当指派一个文件的上下文时，例子同样创建一个流和流句构的上下文。所有的上下文将完全被过滤管理器用 minifilter 提供的回调函数删除。

**passThrough minifilter sample**

描述：passThrough minifilter 说明如何为不同类型的 irp 指定回调函数。

操作原理：passThrough minifilter 没有任何真正的功能。对于每种 I/O 操作,预回调和后回调函数同样被调用，这些回调函数仅仅在堆栈的下一个过滤器之前。

**fastfat file system**

描述: 你可以用它来写全新的文件系统（如果你想的话...）

操作原理：我从来没有考虑过自己写一个新的文件系统，除非boss杀了我...

**scanner minifilter sample**

描述: 本例适用于有志于写检测文件数据过滤器的“Developers”, 尤其是反病毒就是属于这个领域滴（貌似卡巴斯基就用到 minifilter）！

操作原理：scanner 包括用户模式和内核模式组件，内核组件识别合适的时机去扫描一个文件的数据并且传递数据到用户模式组件做更多的确认。用户模式组件创建一个线程的编号去等待内核组件的确认？？在扫描到脏数据字符后，用户模式发送一个合适的反应给内核模式组件。

内核模式组件仅用特定的扩展名（extensions）扫描文件。文件首先在成功的时候扫描。如果文件被写访问打开，扫描将在文件关闭前重新执行。扫描同样可以运行在将被写入到文件的数据上，当数据里发现脏字符串时写将被拒绝。当脏字符在文件关闭期间发现，打印一个 debug 消息。

**swapBuffers minifilter sample**

描述: swapBuffer minifilter 展示如何交换读和写数据之间的缓存。这项技术在加密过滤时相当有用，因为你必须在数据写到磁盘上之前加密，在数据从磁盘上读取后解密。由于加解密必须透明，你不能直接使用系统提供的缓存，因此中间缓存是必须引进滴。

操作原理：swapBuffer minifilter 在读/写或目录控制操作前引进一个新缓存，对应的操作随即运行在新缓存，以代替初始提供的缓存，在操作完成后，新缓存的内容被拷贝回初始缓存。

**MetadataManager**

元数据 I/O 管理，不懂啥意思。

**nullFilter**

没啥功能，只有 minifilter 管理器注册的过程，未实现任何功能，讲解用。

**simrep**

简单演示了，如何通过 PreCreate 回调返回重分析（STATUS\_REPARSE）。该方式允许不打开当前传入路径的文件，而打开另一个路径的文件（重定向）。（目测该功能可用来写沙箱）。实例说明还说被重定向的路径很难被设定为 fast io 方式打开的网络路径（不确定对不对）。

See: <http://blog.csdn.net/jykj_007/article/details/5421397>

See: <http://houyy1990.blog.163.com/blog/static/603109892012105583253/>