> 背景
>
> 已知`Coresight`设备是在`etb/etr`中设计了一个`ring buffer`，将trace 得到的perf_event存入`ring-buffer`（如何从中读数据还挺细节，涉及什么overwrite，估计是`ring-buffer`对trace设备是透明的，以致于有可能覆盖`ring-buffer`的头指针）。
>
> perf会在用户态也创建一个`ring-buffer`并与内核态（`etb/etr`）的`ring-buffer`形成`mmap`，这样perf就在用户态直接提取到各个perf_event,然后塞到`perf.data`中

工作目标

寻找合适的解析`perf.data`的工具，提取出程序崩溃前`ip`寄存器所存数据的变化，或者说程序崩溃前的指令流。

5.2进度

昨天根据官方`perf-data-format.txt`中的提示找到了一个用于解析`perf.data`的python工具——`pmutools`，他用了python的`construct`包，因此更清晰地展示了`perf.data`的的二进制结构（这个包很适合处理二进制文件）。

通过它的代码("`/parser/perfdata.py`")，我进一步摸清了`perf.data`的的二进制结构（迫于`perf-data-format.txt`写的太烂，简直human unreadable）。但后来发现这个工具其实没有写完，而且他不能解析龙哥之前跑`latex2rtf`得到的`perf.data`（工具会报错，估计就是没写完）。所以它并不能用。后来发现搞清楚`perf.data`的结构其实用处并不大，毕竟我们的目标是perf_event里的`ip`。

只能转过头来研究`perf script`了，毕竟他还是perf官方的工具，但是是用`cpp`写的，读源代码是不可能的。不过工具很好用，能直接得到`ip`，但是感觉把他输出到`stdout`或者到其他文件再读进自己的工具比较不优雅（除非改源代码呜呜呜）。还有一个问题就是，解析出来的`ip`里一开始会有大量的重复`ip`，目前还是没摸清楚这是为什么。

5.3进度

咕咕咕