**问题总结**

1. 冻屏问题  
   多线程共享egl环境下出现，某个渲染线程退出时调用eglTermilate，导致还在运行的线程调用gpu接口出现寻址错误 smmu fault，而8295上出现smmu fault，smmu模块会dump pagetable，这个过程会加锁，导致weston的请求hgsl的hab请求block，从而整个系统冻住。  
   分析过程：  
   点击事件卡在weston未下发，weston卡住了，从卡住的堆栈上看到hgsl\_hyp\_mem\_map\_smmu 流程卡住，看qnx时间的日志，发现smmu\_service检测到GPU的smmu fault，并看到kgsl 一直处于死锁状态，所以weston的smmu请求block了，从而系统卡住了；什么原因导致smmu fault的呢，从场景看都是page进程退出时发生的，分析日志发现，agil渲染线程先退出的，他调用了eglTerminate ，但是后面webengine线程还在swapbuffers，找到问题原因了，解决办法是egl进行wrap，将eglInitialize和eglTerminate接口进行计数，调用时判断计数，计数为0才，真正释放，否则只是修改计数器  
     
     
   2024-08-30 16:24:48.646 45069 6 smmu\_service.45069 default 2 smmu\_service[smmu.c:551]: SMMU:FAULT on Device GPU, CB num 0(HLOS)  
   2024-08-30 16:24:48.647 86053 20 KGSL.86053 OS-Buffer 1 KGSL:(tid:20):(caller\_pid:86053) {0x74722000|0x3002F003} FATAL ERROR in rgs\_smmu\_fault\_callback[3313]: fault pid 10313851, process name 'citydriv'  
     
   2024-08-15 23:27:30.106 86053 1 kgsl.86053 slog 5 #2 (86053:20) potential deadlock while trying to lock mutex rgs\_pmem\_table\_mutex[0x12eeb1a180] in rgs\_pers\_lock\_pmem\_tbl\_mtx:913, previously locked in rgs\_pers\_lock\_pmem\_tbl\_mtx:913 by thread 32   
   2024-08-15 23:27:30.106 86053 1 kgsl.86053 slog 5 #2 (86053:33) potential deadlock while trying to lock mutex rgs\_pmem\_table\_mutex[0x12eeb1a180] in rgs\_pers\_lock\_pmem\_tbl\_mtx:913, previously locked in rgs\_pers\_lock\_pmem\_tbl\_mtx:913 by thread 32   
   2024-08-15 23:27:30.109 86053 1 kgsl.86053 slog 5 #2 (86053:35) potential deadlock while trying to lock mutex rgs\_pmem\_table\_mutex[0x12eeb1a180] in rgs\_pers\_lock\_pmem\_tbl\_mtx:913, previously locked in rgs\_pers\_lock\_pmem\_tbl\_mtx:913 by thread 32   
   2024-08-15 23:27:30.137 86053 1 kgsl.86053 slog 5 #2 (86053:26) potential deadlock while trying to lock mutex rgs\_pmem\_table\_mutex[0x12eeb1a180] in rgs\_pers\_lock\_pmem\_tbl\_mtx:913, previously locked in rgs\_pers\_lock\_pmem\_tbl\_mtx:913 by thread 32   
   2024-08-15 23:27:30.204 86053 1 kgsl.86053 slog 5 #2 (86053:31) potential deadlock while trying to lock mutex rgs\_pmem\_table\_mutex[0x12eeb1a180] in rgs\_pers\_lock\_pmem\_tbl\_mtx:913, previously locked in rgs\_pers\_lock\_pmem\_tbl\_mtx:913 by thread 32   
   2024-08-15 23:27:52.856 86053 1 kgsl.86053 slog 5 #2 (86053:32) unlocking mutex 0x12eeb1a180 previously locked in rgs\_pers\_lock\_pmem\_tbl\_mtx:913 by thread 32  
   看qnx日志，kgsl线程出现了死锁，直到 线程(86053:32) unlocking mute释放锁才恢复正常  
     
     
   要解决这个问题，需要处理多个线程共享egl环境时，退出资源释放问题，在  
   <https://yuque.alibaba-inc.com/agil/tg1v4v/sgfewxpensdtwb50>

NS2 调用了eglTerminate之后

WebEngine还在swapbuffer

webengine相当于使用了已经被清理的资源。

那么这块有一些关于egl的技术问题，需要补充一下相关知识：

* 为什么NS2 调用了eglTerminate之后，webengine就不能继续swapbuffer？
* 答：eglTerminate会把进程上所有关联在EGLDisplay上的资源释放掉。本问题里的情况是ns2这边创建了egl环境和一个EGLContext (ctx)，然后这个ctx共享给webengine，webengine从而可以共享使用ns2这边的opengl和egl资源与环境。而webengine的swapbuffer会使用EGLSurface，这是关联在EGLDisplay上的，所以ns2这边terminate后，所有egl与opengl环境资源释放，webengine这边再swapbuffer会调用已经释放的egl对象，从而产生错误。

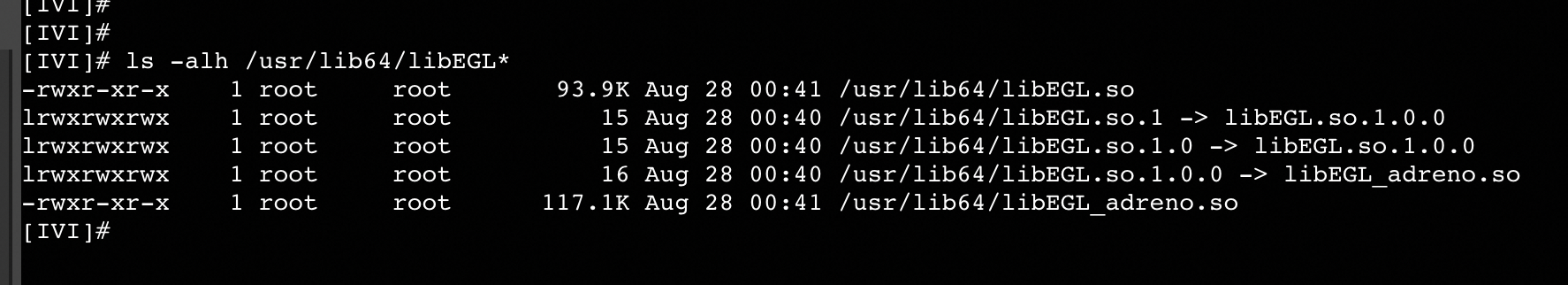
## eglWrapper 这个模块

上面大量的讲到了egl-api，我们系统里，由图形组（weston组）提供了一个eglWrapper，实际上要求每个渲染引擎在调用egl-api的时候，都要走这个eglWrapper，历史原因麻烦这里填充一下相关原因：

如前所述，经常存在这种agile和webengine都要操作同一个shared context的场景，为了避免多线程操作出现问题，也就是线程A执行了eglTerminate（）导致连接被关闭，线程B再去访问数据出现异常，我们给libEGL.so增加了一个wrapper层，给EGL\_Display增加引用计数，eglnitialize()的时候对引用计数inc, eglTerminate()的时候去对引用计数dec，在引用计数减少到0的时候，才去执行GPU驱动的eglTerminate()操作，这就要求所有的模块都通过wrapper来调用libEGL.so，如果有模块bypass了，引用计数可能就出现问题。

## webengine为什么没走wrapper

根据排查，webengine用到的egl-api，实际上没走wrapper，导致了一些问题，麻烦解答一下，关于库的链接问题，为什么webengine没有走到wrapper：



如图所示：webengine调用的libEGL.so.1这个库来初始化egl环境，进而调用高通的库，但是weston提供的wrapper库是libEGL.so，需要weston提供一个标准的linux命名的so来使引擎自动调用到weston的wrapper库

## CAF的多窗口机制

在CAF应用下，多个窗口，虽然是多个渲染线程，但 eglInitialize 这个接口只调用了一次，就是第一个window创建的时候，去调用了一次。

后续有新window创建的时候，eglInitialize 没有调用，说明复用了一些 egl 环境。

这与 NS2 不同。NS2 是 每一个window，都会调用一次 eglInitialize。所以 NS2 里每一次window 销毁的时候，都要调用一次 eglTerminate。

## CAF+webengine的机制

CAF启动的时候，会自己调用一次 eglInitialize， 然后 webengine 初始化的时候，也会自己调用一次 eglInitialize。

但是进程退出的时候，CAF 确实调用了 eglTerminate，但是 webengine 是没有调用 eglTerminate。

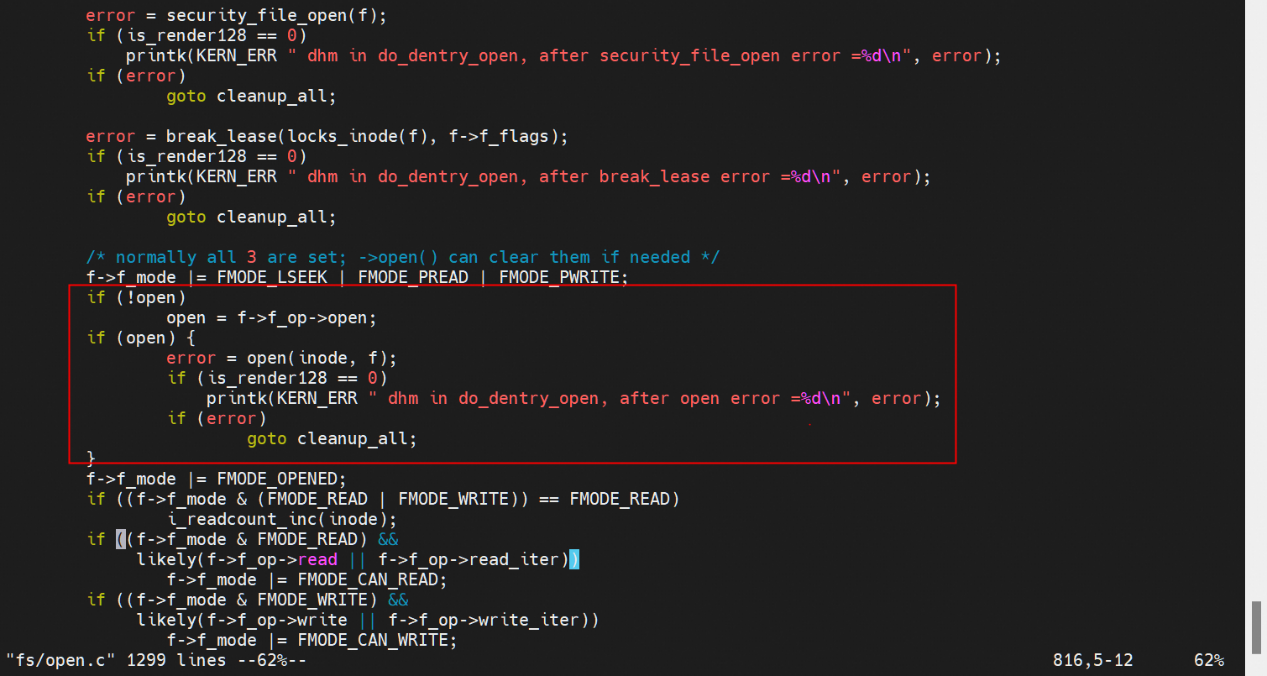
所以，进程最终退出之后，egl 环境实际上是没有清理的，导致偶现的，在主线程会 crash 在 libGLESv2\_adreno。

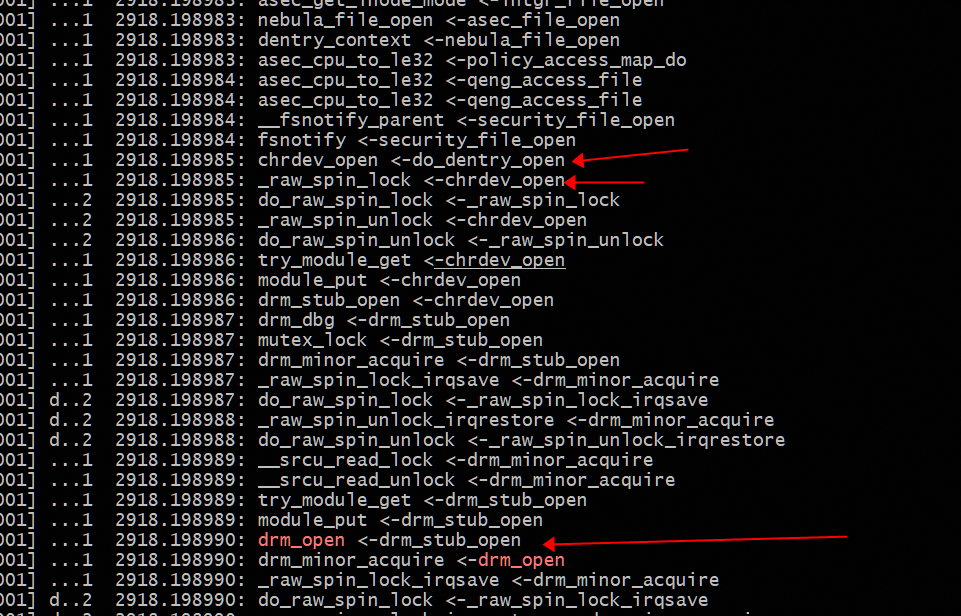
## 此问题的最终解法

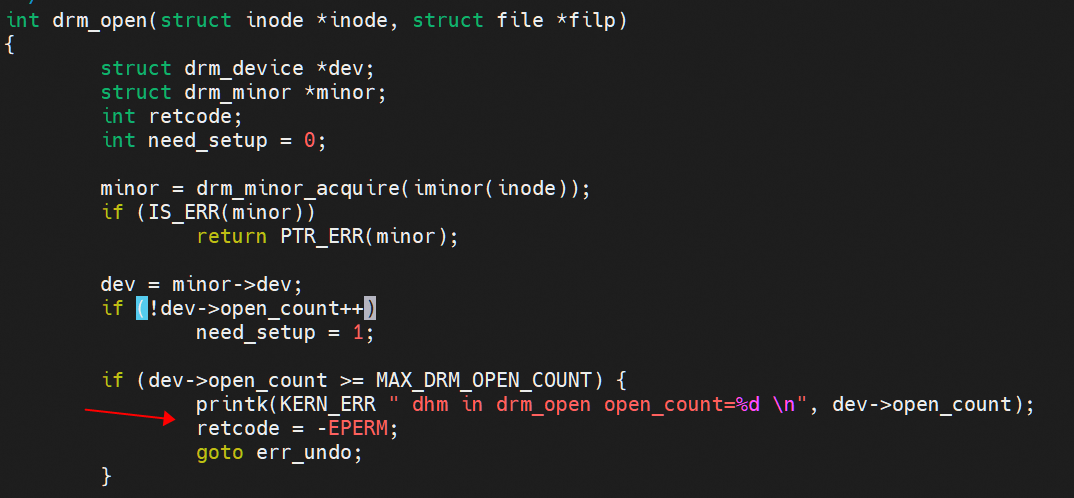
谁调用了 eglInitialize

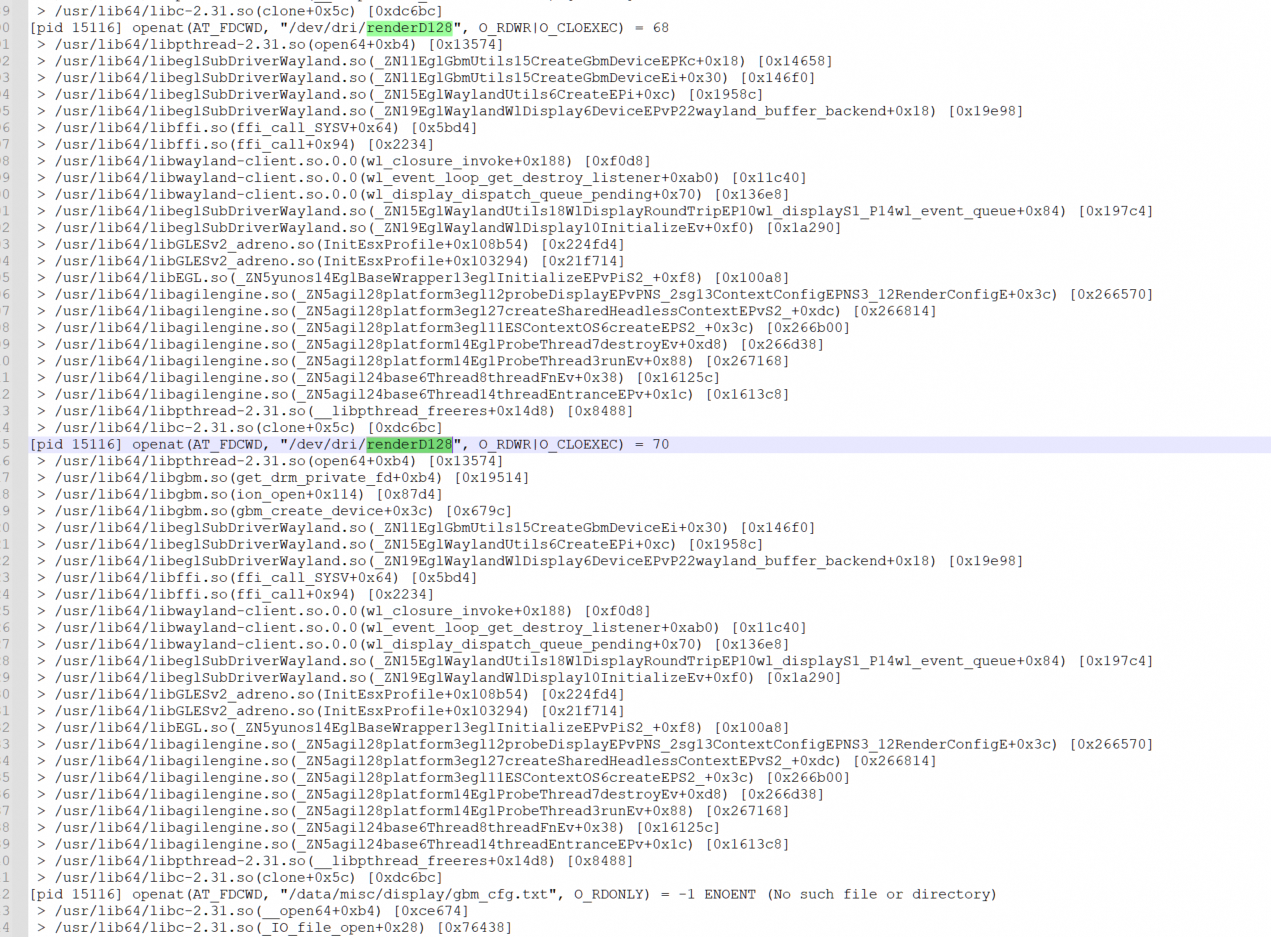
谁就要调 eglTerminate

1. eglInitialize failed EPERM  
   已经找到问题原因：在openat流程打日志定位到问题fs/open.c的f->f\_op->open 函数这里返回了EPERM

  
通过ftrace找到open对应的函数是drm\_open



[vendor](http://11.165.236.46:36666/P12L_baseos_sa8295_dev/xref/vendor/)/[qcom](http://11.165.236.46:36666/P12L_baseos_sa8295_dev/xref/vendor/qcom/)/[bsp](http://11.165.236.46:36666/P12L_baseos_sa8295_dev/xref/vendor/qcom/bsp/)/[linux](http://11.165.236.46:36666/P12L_baseos_sa8295_dev/xref/vendor/qcom/bsp/linux/)/[drivers](http://11.165.236.46:36666/P12L_baseos_sa8295_dev/xref/vendor/qcom/bsp/linux/drivers/)/[gpu](http://11.165.236.46:36666/P12L_baseos_sa8295_dev/xref/vendor/qcom/bsp/linux/drivers/gpu/)/[drm](http://11.165.236.46:36666/P12L_baseos_sa8295_dev/xref/vendor/qcom/bsp/linux/drivers/gpu/drm/)/[drm\_file.c](http://11.165.236.46:36666/P12L_baseos_sa8295_dev/xref/vendor/qcom/bsp/linux/drivers/gpu/drm/drm_file.c)  
在drm\_open函数的dev->open\_count >= MAX\_DRM\_OPEN\_COUNT条件加日志，再次复现问题，发现这里超出了128限制，返回了EPERM  


strace工具，抓到一个进程最少打开2个/dev/dri/renderD128节点的堆栈，与8155只打开一个有不同  


<https://aone.alibaba-inc.com/v2/project/2079467/bug/58822252>  
  
3、回溯栈踩坏  
crash问题，想用asan跑出踩内存原因，dlopen libllvm-qcom.so 有RTLD\_DEEPBIND flag导致asan失败，需要找到代码dlopen的地方，但是源码搜不到，通过strace抓到打开的库是libgsl.so，高通的库，没有源码，尝试自己修改里面的值。  
  
用strace抓到dlopen打开libllvm-qcom.so库的调用栈

openat(AT\_FDCWD, "/usr/lib64/libllvm-qcom.so", O\_RDONLY|O\_CLOEXEC) = 202

 > /usr/lib/ld-2.31.so(\_dl\_catch\_error+0x1264) [0x17c64]

 > /usr/lib/ld-2.31.so() [0x6db0]

 > /usr/lib/ld-2.31.so() [0x74b4]

 > /usr/lib/ld-2.31.so() [0x879c]

 > /usr/lib/ld-2.31.so(\_dl\_find\_dso\_for\_object+0x5c0) [0x12410]

 > /usr/lib64/libc-2.31.so(\_dl\_catch\_exception+0x58) [0x113cc8]

 > /usr/lib/ld-2.31.so(\_dl\_find\_dso\_for\_object+0x188) [0x11fd8]

 > /usr/lib64/libdl-2.31.so() [0x11ac]

 > /usr/lib64/libc-2.31.so(\_dl\_catch\_exception+0x58) [0x113cc8]

 > /usr/lib64/libc-2.31.so(\_dl\_catch\_error+0x24) [0x113d64]

 > /usr/lib64/libdl-2.31.so(dlerror+0x2dc) [0x187c]

 > /usr/lib64/libdl-2.31.so(dlopen+0x38) [0x1238]

 > /usr/lib64/libgsl.so(os\_lib\_map+0x14) [0x1c734]

 > /usr/lib64/libCB.so(cl\_common\_is\_half\_subnormal+0x34) [0x4eb24]

 > /usr/lib64/libCB.so(cl\_compiler\_wrapper\_handle\_from\_binary+0xa4) [0x4f9f4]

 > /usr/lib64/libCB.so(cl\_a6x\_create\_utility\_programs+0x17c) [0xb0ffc]

 > /usr/lib64/libCB.so(cl\_a6x\_initialize\_global\_gsl\_state+0x38) [0x9e618]

 > /usr/lib64/libCB.so(cb\_common\_initialize\_global\_gsl\_state+0x50) [0x4e200]

 > /usr/lib64/libCB.so(cb\_create\_context+0x350) [0x511e0]

 > /usr/lib64/libOpenCL.so(qCLDrvAPI\_clCreateContext+0x84) [0xb284]

 > /usr/lib64/libheidi.so(\_ZNSbIDiSt11char\_traitsIDiESaIDiEE4\_Rep8\_M\_cloneERKS1\_m+0x944014) [0x18899f4]

 > /usr/lib64/libheidi.so(\_ZNSbIDiSt11char\_traitsIDiESaIDiEE4\_Rep8\_M\_cloneERKS1\_m+0x93c290) [0x1881c70]

 > /usr/lib64/libheidi.so(\_ZNSbIDiSt11char\_traitsIDiESaIDiEE4\_Rep8\_M\_cloneERKS1\_m+0x92e424) [0x1873e04]

 > /usr/lib64/libheidi.so(\_ZNSbIDiSt11char\_traitsIDiESaIDiEE4\_Rep8\_M\_cloneERKS1\_m+0x19bb5c) [0x10e153c]

 > /usr/lib64/libheidi.so(bef\_ai\_image\_quality\_enhancement\_video\_lite\_hdr\_destory+0x88) [0x8bdfe8]

 > /usr/lib64/libheidi.so(\_ZNSt8\_\_detail16\_Hashtable\_allocISaINS\_10\_Hash\_nodeISt4pairIKSsfELb1EEEEE16\_M\_allocate\_nodeIJRS3\_RiEEEPS5\_DpOT\_+0x48c) [0x8bc12c]

 > /usr/lib64/libheidi.so(bef\_ai\_image\_quality\_enhancement\_video\_lite\_hdr\_create+0x68) [0x8bdc68]

 > /usr/lib64/cow/libVideoFilterLens.so(\_ZN8YUNOS\_MM15VideoFilterLens7initSdkEv+0x274) [0x10f04]

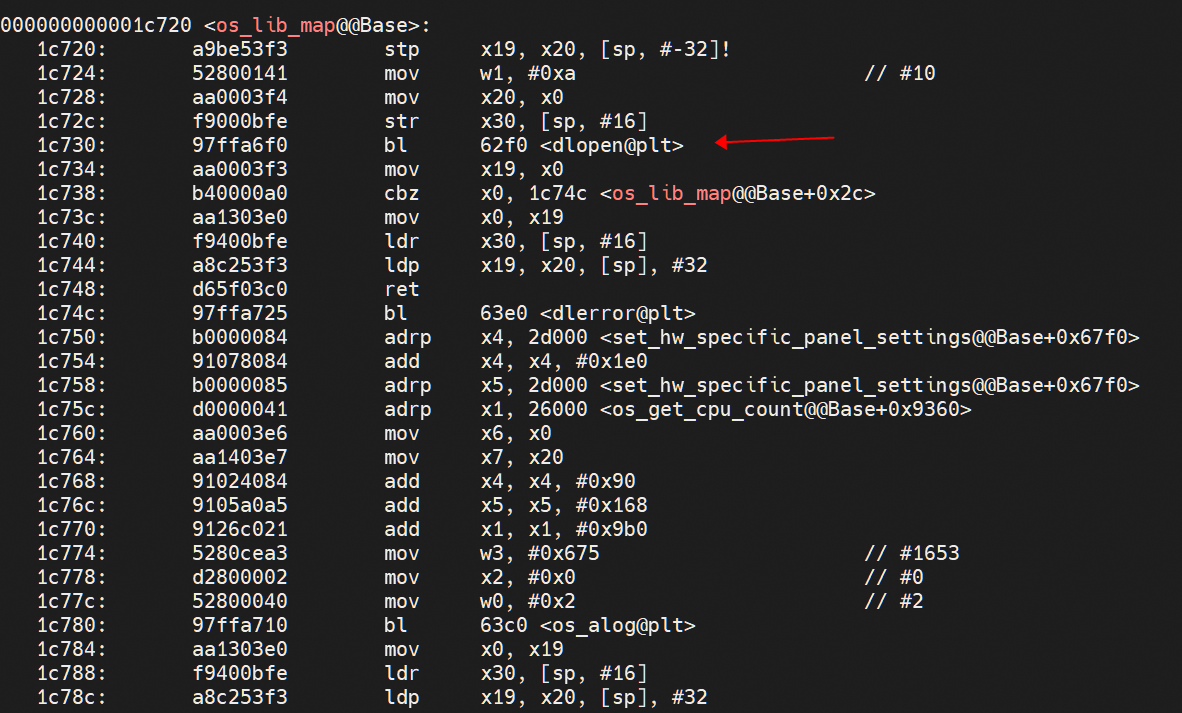
 > /usr/lib64/cow/libVideoFilterLens.so(\_ZN8YUNOS\_MM14LensPollThread4mainEv+0xb44) [0xc62c]

 > /usr/lib64/libmmbase.so(\_ZN8YUNOS\_MM8MMThread10threadfuncEPv+0x68) [0x2fcb0]

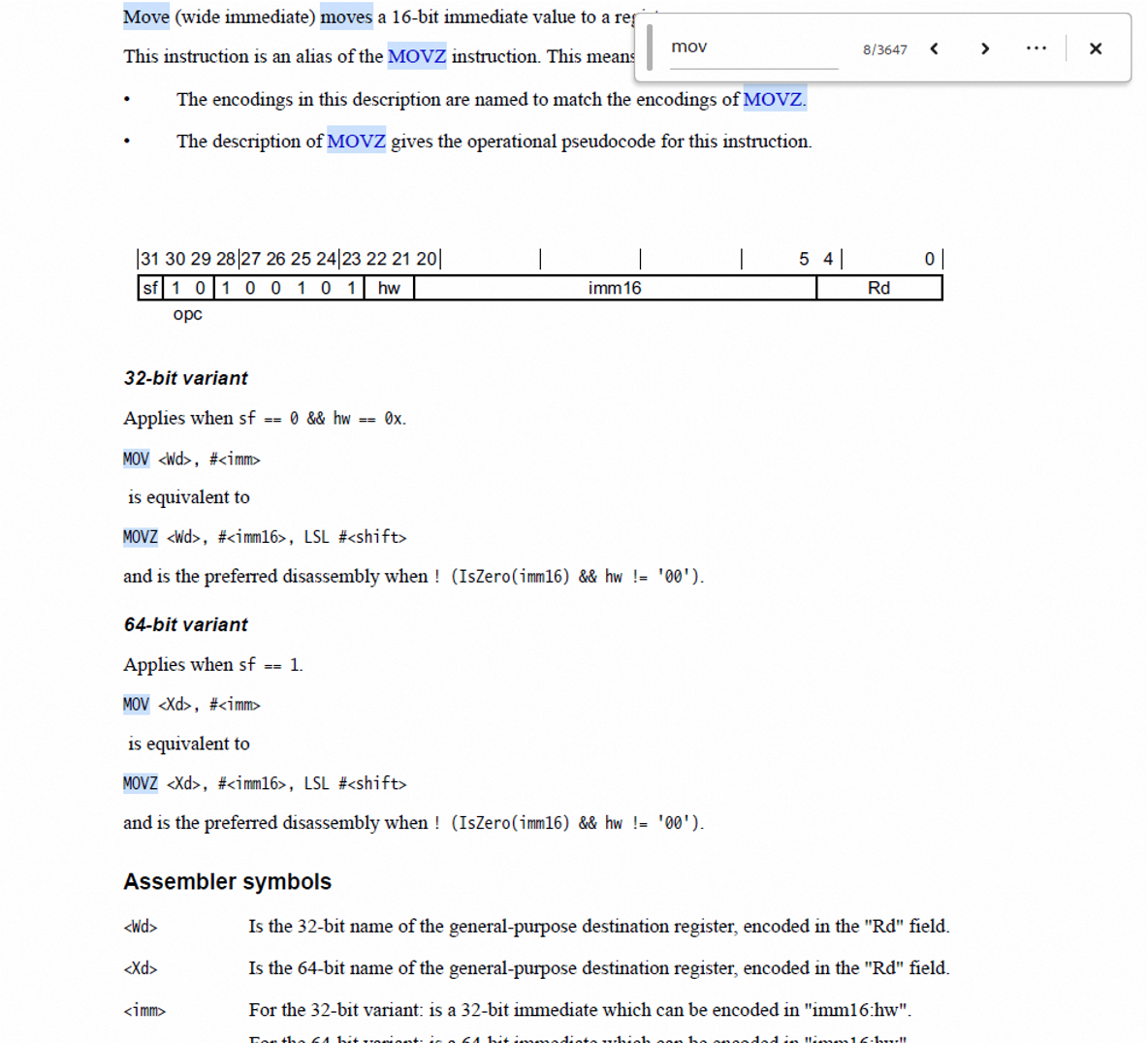
 > /usr/lib64/libpthread-2.31.so(\_\_libpthread\_freeres+0x14d8) [0x8488]

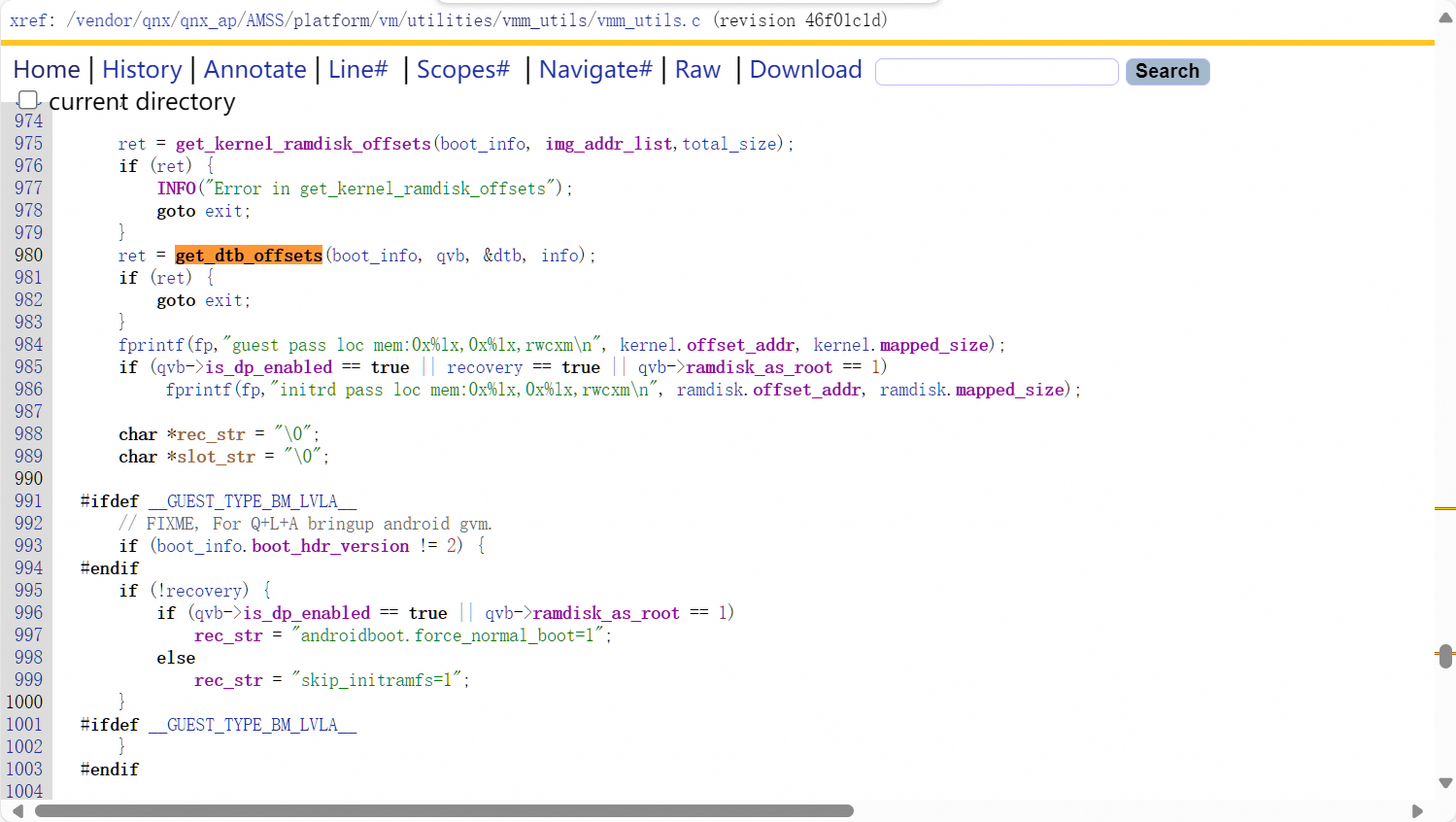
 > /usr/lib64/libc-2.31.so(clone+0x5c) [0xdc6bc]

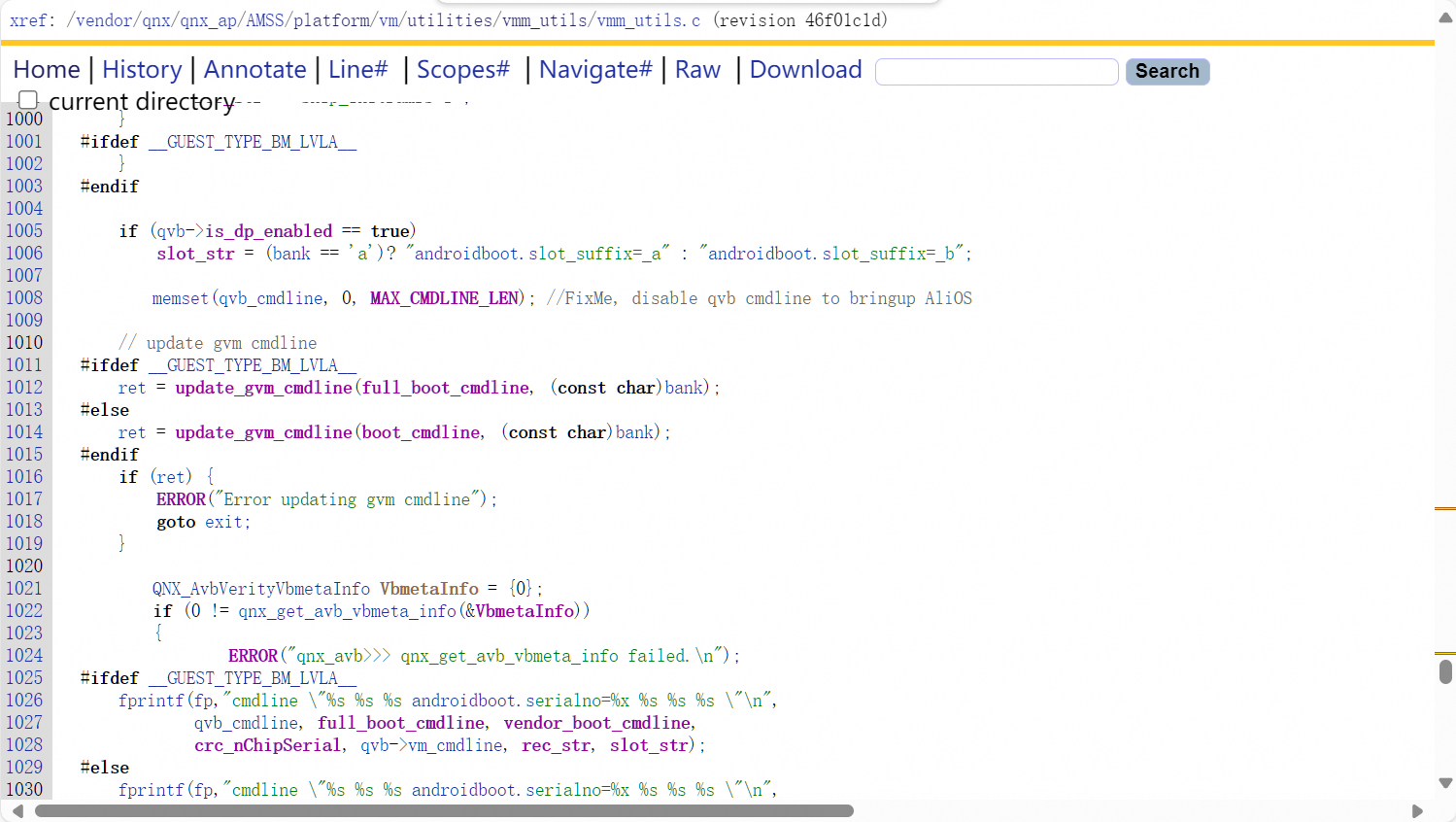
反汇编libgsl.so，找到strace中的os\_lib\_map函数，看到是0x000000000001c724 <+4>: mov w1, #0xa // #10  
52800141 对应的字段分解如

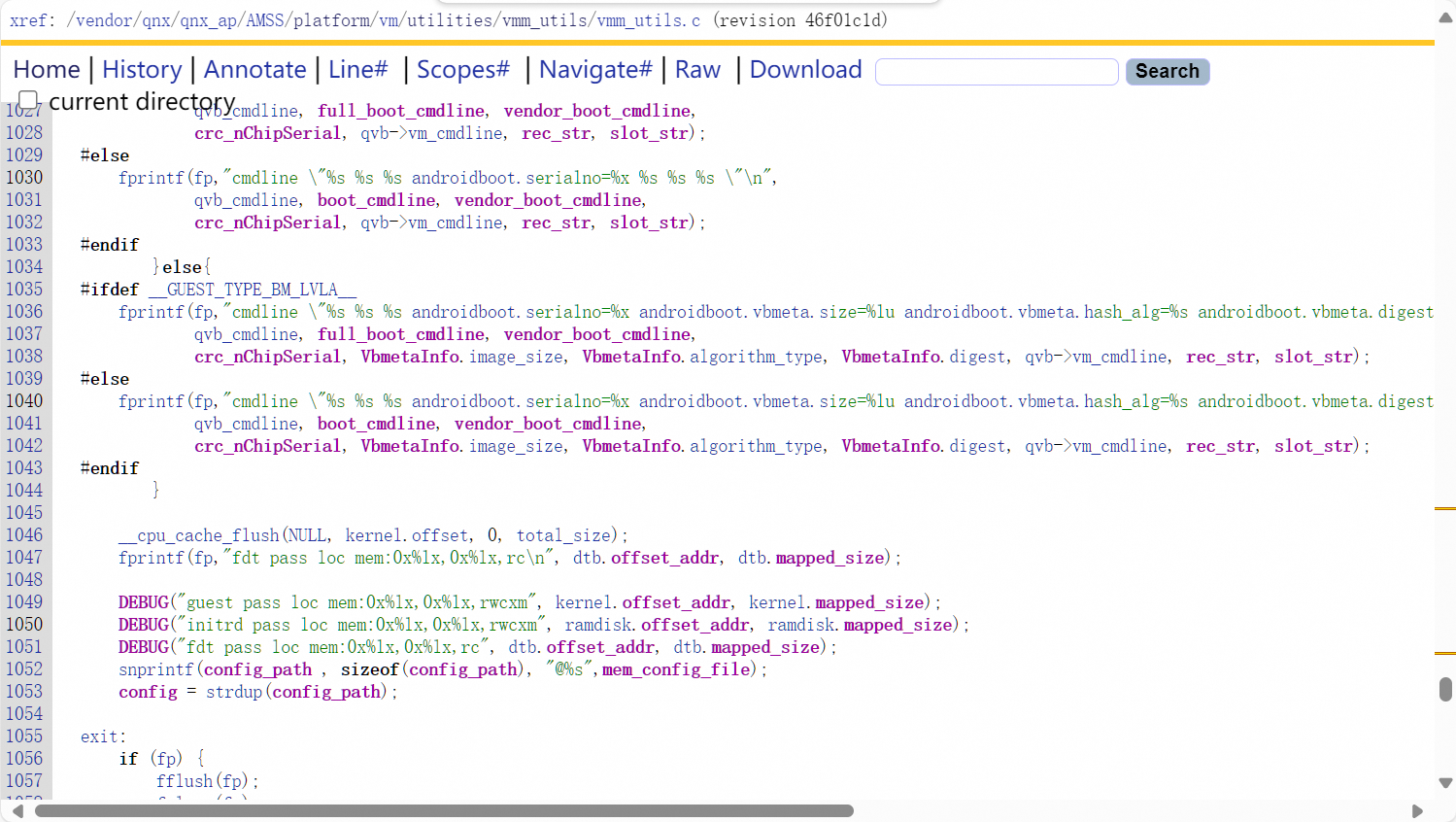


对照arm64 mov指令将52800141 展开成二进制，可以发现汇编指令的10，对应的字段，改成8即可

  
  
这样asan能跑起来，但是没有抓到踩内存的问题  
换一种方法，c++ 嵌入汇编命令 asm mov将x29寄存器，也就是sp寄存器保存到内存，使用mprotect栈保护系统调用，将栈保护起来，为了防止后面子函数的误报，申请一段栈内存，防止子函数返回对栈的修改，这样再测试跑出问题  
<https://blog.csdn.net/tyyj90/article/details/130024069>  
  
4、黑屏问题  
A、车机adb无法识别，串口可以，通过串口日志抓到slog2info日志，看到qvm进程未拉起，进一步分析slog日志，看到启动流程停止了，结合qnx源码：  
从代码流程看，get\_dtb\_offsets 返回应该是正确的0，回到get\_config\_linux，然后紧接着执行fprintf将内容写到/dev/shmem/linux-lv-mem.config，后面没有日志，不确定是写失败，还是 fflush(fp); 没有成功





  
从问题车机看linux-lv-mem.config内容是空的：  
# cat /dev/shmem/linux-lv-mem.config   
#   
从正常车机看：  
# cat /dev/shmem/linux-lv-mem.config   
guest pass loc mem:0xe0080000,0x2e21000,rwcxm   
cmdline " root=/dev/vda14 rootwait ro rootfstype=ext4 init=/usr/lib64/systemd/systemd-bootchart loglevel=4 kdbus aliosboot.slot\_suffix=\_a androidboot.serialno=2248e6c2 console=ttyAMA0 user\_debug=31 print-fatal-signals=1 no\_console\_suspend firmware\_class.path=/firmware/image/ fsck.repair=yes swiotlb=5120 kpti=1 pcie\_ports=compat skip\_initramfs=1 "   
fdt pass loc mem:0x3ede85000,0x35f52,rc  
正常车机上看，这是alios镜像文件的分区  
[IVI]# partx /dev/vda14   
NR START END SECTORS SIZE NAME UUID   
14 21836208 30224815 8388608 4G ivi\_system\_a 387925ee-1fe5-7a76-7248-2d5b637f3f65  
/dev/shmem/linux-lv-mem.config是空的，这是vmm\_service拉起qvm进程传入的参数，用来配置alios镜像加载分区，内存地址等等信息，文件为空alios镜像分区都不读不到，无法加载alios镜像文件  
另外，串口进去很多命令都会卡住，比如df 、mount、cat 文件等。  
从抓的slog可以看到一些ufs的错误，不确定是不是与这有关： Jan 01 00:00:30.170 devb\_ufs\_qualcomm.86059 slog 100 ufshc\_reg\_wait: timeout (1000000us) reg 0x2458ba5078, mask 0x1, wval 0x0, val 0x1 Jan 01 00:00:30.170 devb\_ufs\_qualcomm.86059 slog 100 ufshc\_reg\_wait: fast poll done (1000000us) reg 0x58ba5078, mask 0x1, wval 0x0, val 0x0, cnt 1000 Jan 01 00:00:30.170 devb\_ufs\_qualcomm.86059 slog 100 ufshc\_reg\_wait: fast poll done (1000000us) reg 0x58ba5058, mask 0x8, wval 0x0, val 0x0, cnt 1000 Jan 01 00:00:30.170 devb\_ufs\_qualcomm.86059 slog 100 ufs\_timer: timeout (ptl) 0:0:6 Jan 01 00:00:30.170 devb\_ufs\_qualcomm.86059 slog 100 ufshc\_abort: UFSHC\_UTRLDBR indicates req cmplt but ocs 0xf  
  
B、车机adb无法识别，串口可以大量服务挂掉  
从串口日志看，vda22对应/keyinfo分区，/keyinfo分区mount失败导致weston/audio等系统服务挂了，界面黑屏  
[ 2.134917] EXT4-fs error (device vda22): ext4\_lookup:1789: inode #28: comm chown: deleted inode referenced: 113   
[ 2.155939] Aborting journal on device vda22-8.   
[ 2.174724] EXT4-fs (vda22): Remounting filesystem read-only   
[ 2.174738] EXT4-fs error (device vda22): ext4\_journal\_check\_start:61: Detected aborted journal   
[ 2.175154] EXT4-fs error (device vda22): ext4\_lookup:1789: inode #28: comm chown: deleted inode referenced: 114  
串口查看qnx的mount情况，出现卡住，而且/var分区挂载也是只读的ro，可能是硬件出现了问题，导致很多分区未mount成功，请零束帮忙看下

使用dd命令将vda22分区写成0能复现问题  
<https://aone.alibaba-inc.com/v2/project/2077592/bug/59376281>  
<https://aone.alibaba-inc.com/v2/project/2077592/bug/58219146>