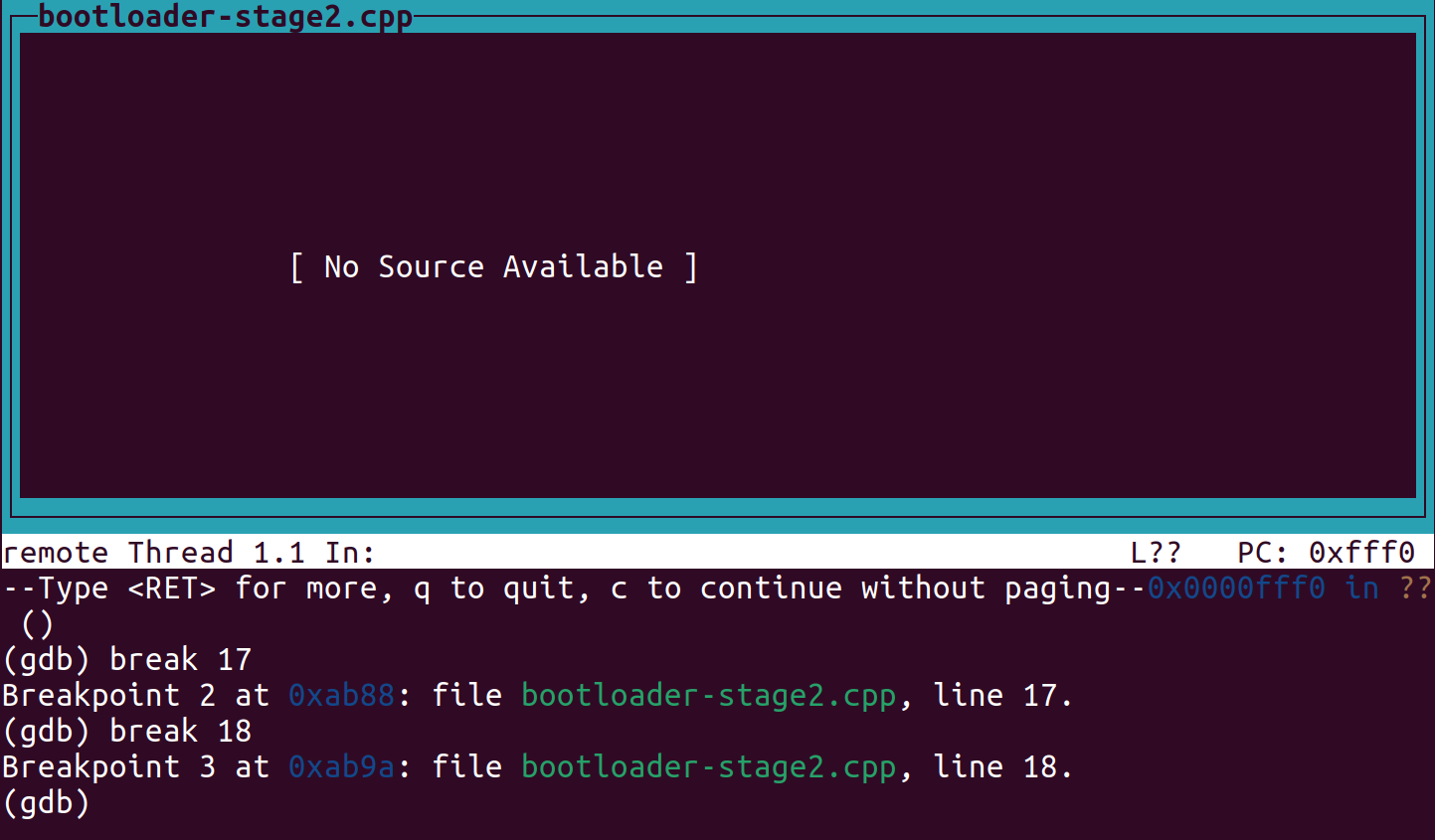
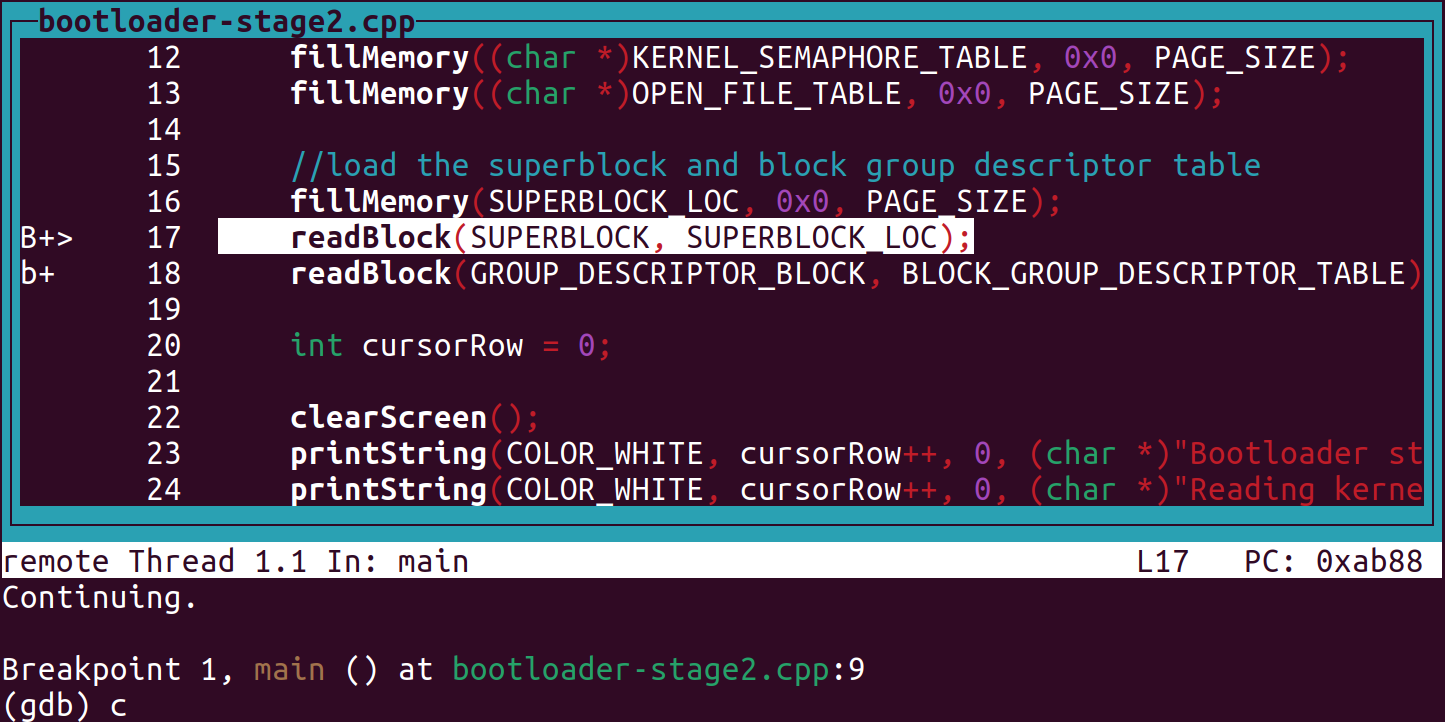


Press Enter

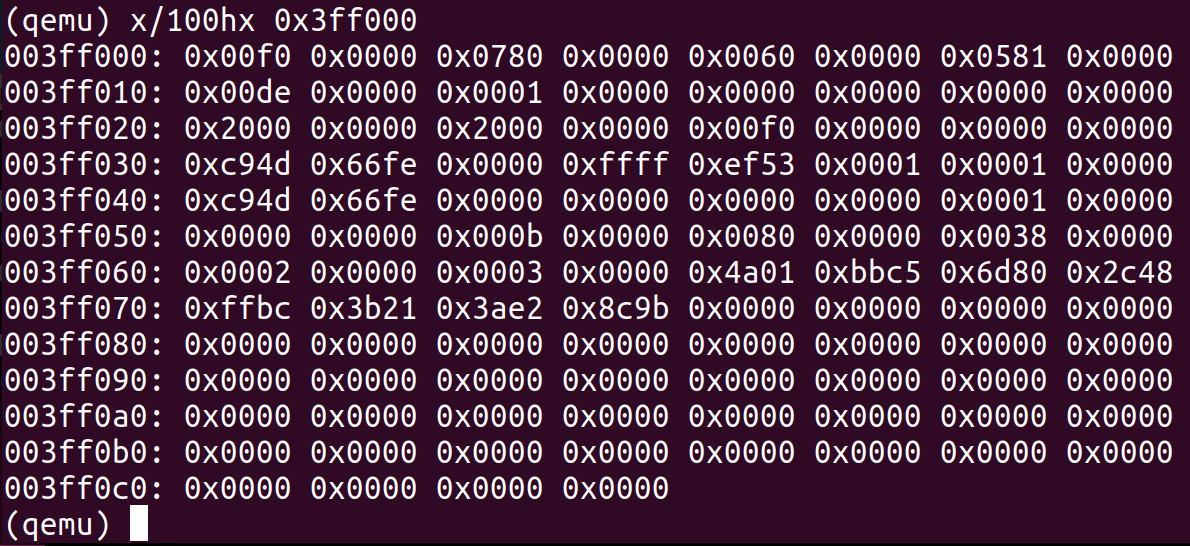


Set breakpoints at line 17 & 18



Type “c” to continue to the breakpoint at line 17 (where we test readBlock)

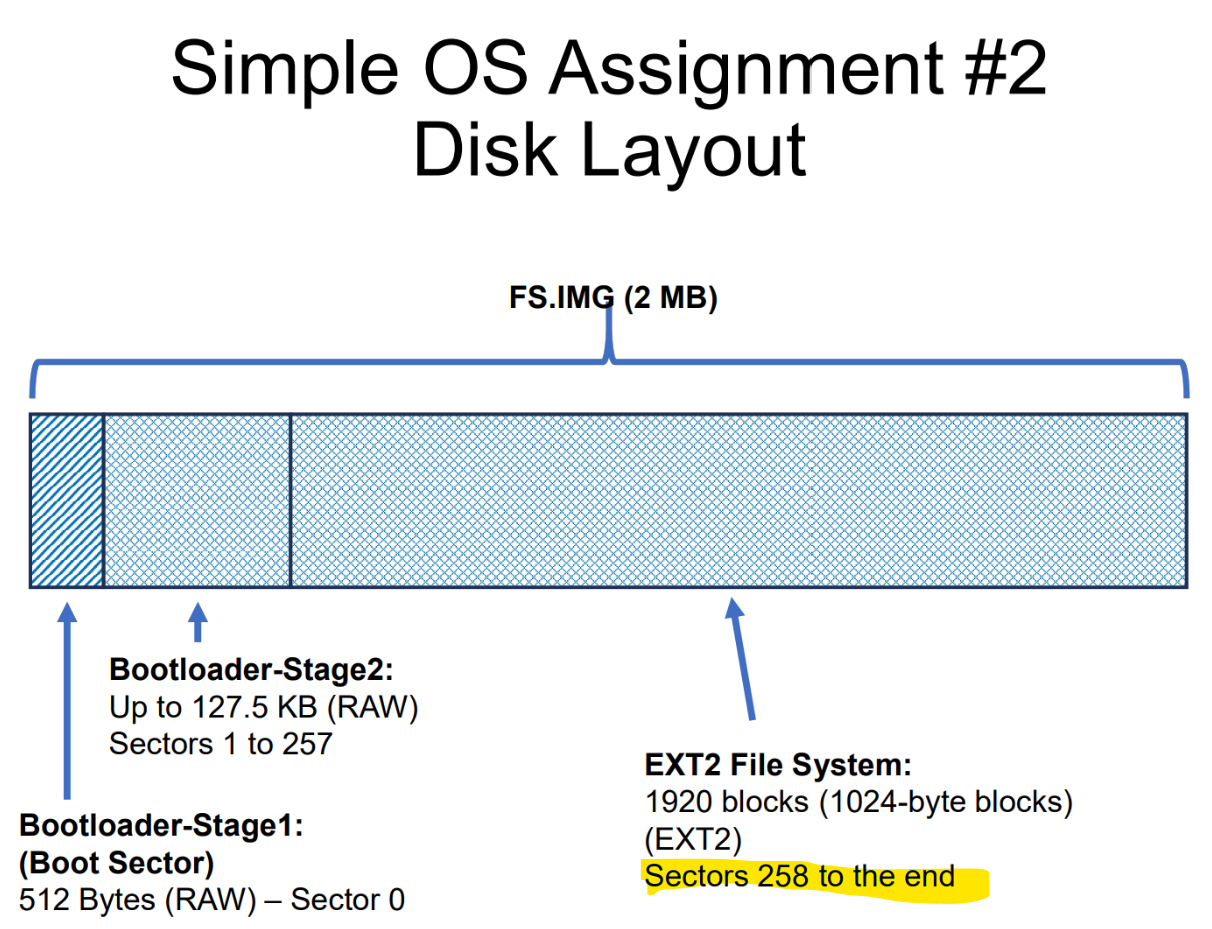
Go back to the QEMU terminal and type this command to analyze the SuperBlock memory location (where the readBlock reads to)



Open up a new terminal in the SimpleOS folder to load the kernel files

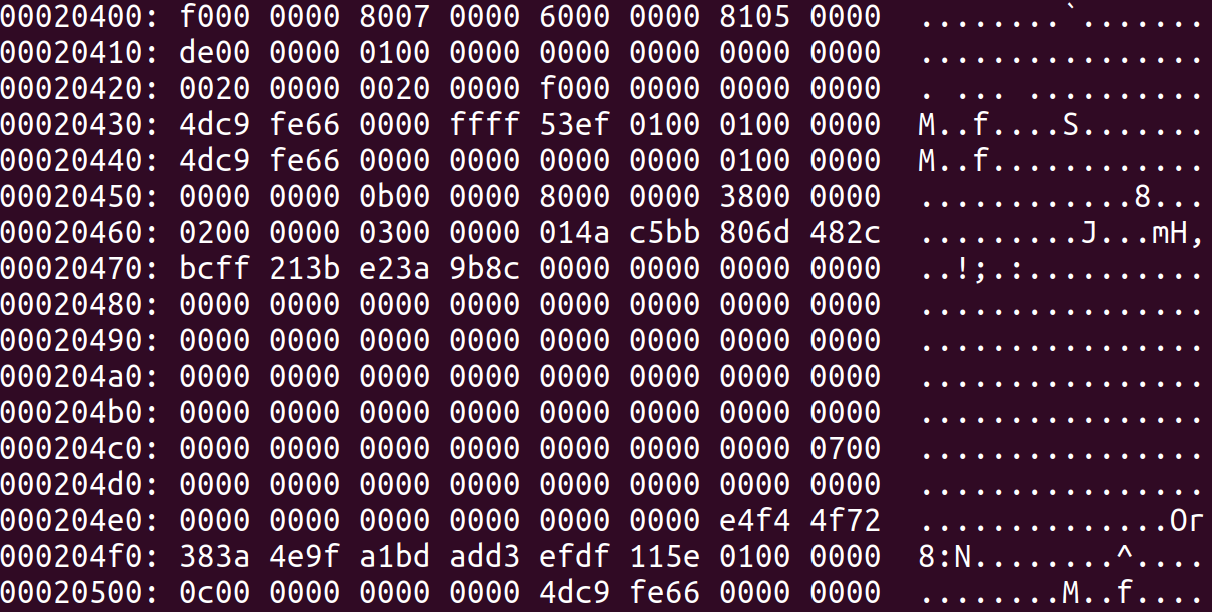


Testing diskReadSector(can’t test diskWriteSector, but the only difference is one global variable):



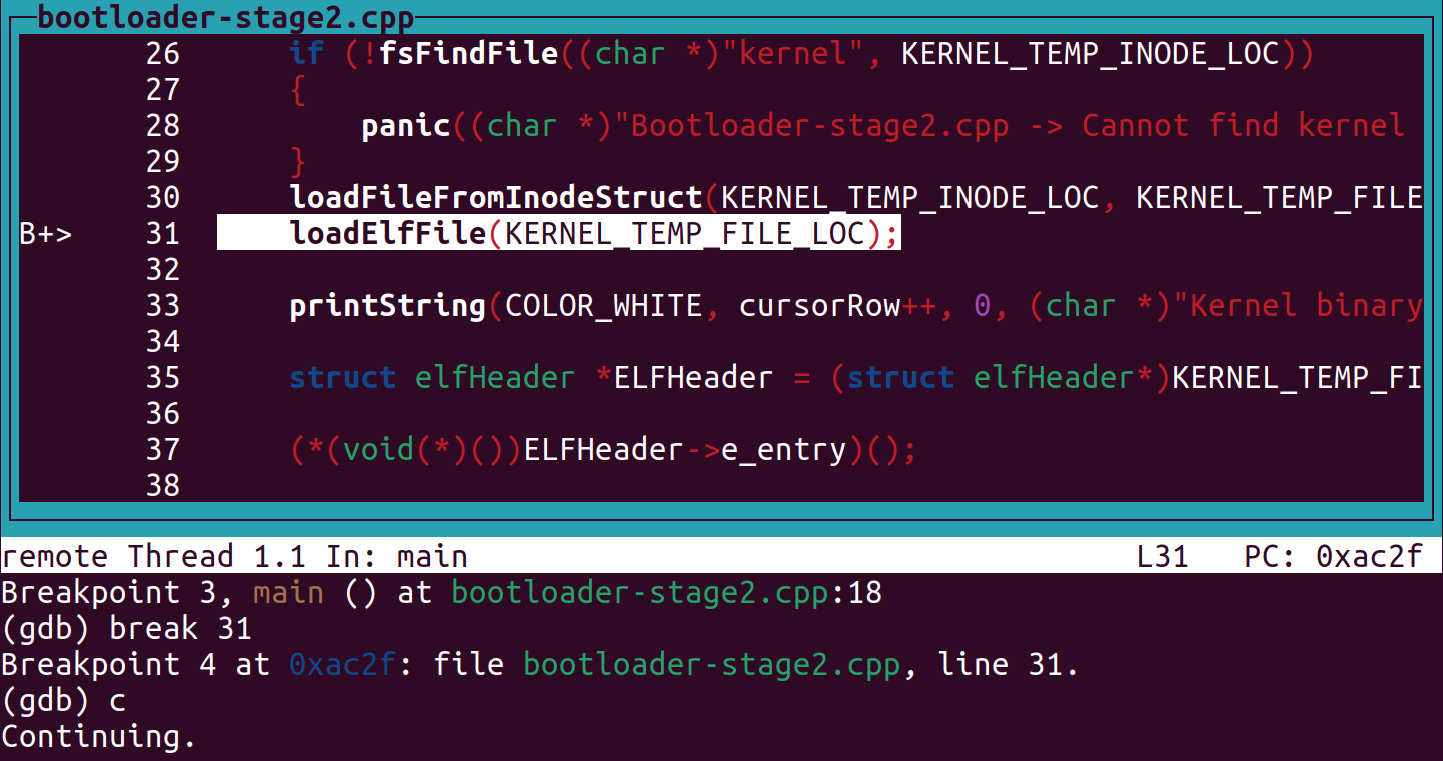
Superblock is at the beginning of EXT2 File System which is in sector 258

Each sector is 512 bytes, so 258 \* 512 = 132096 (**20400 in HEX**)



This one is in Little Endian format, so unless you want to convert it just look for matching numbers that are in different orders in the QEMU screenshot above. If they are similar then the readBlock is correct

Testing loadFileFromInodeStruct:

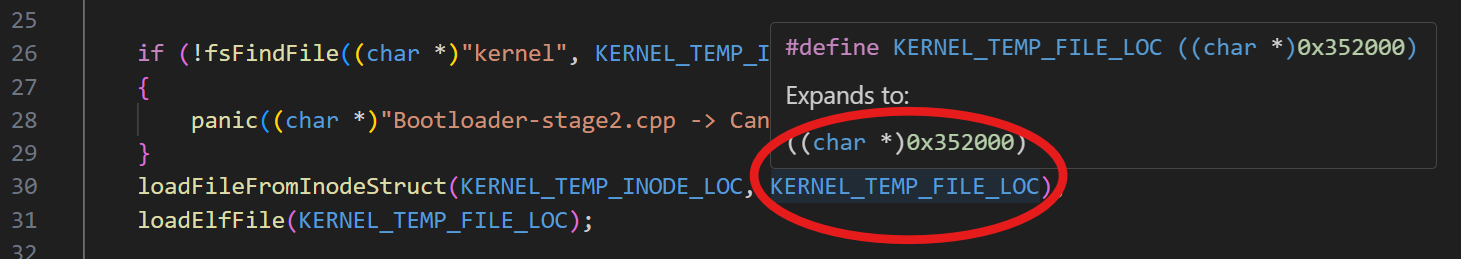


Set a breakpoint at line 31 and continue



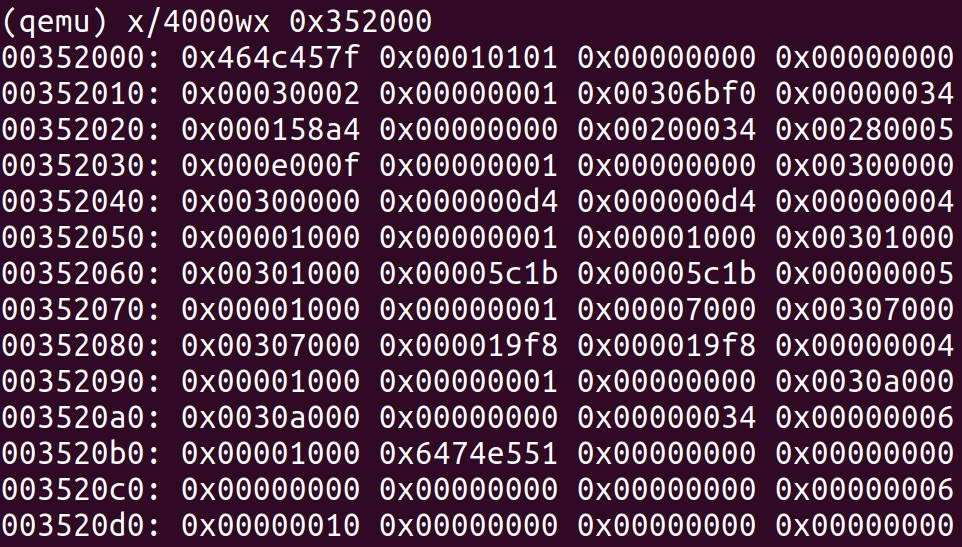
The kernel file is what you are loading so it will have this data no matter what. The following screenshots are of the QEMU window in your GDB debugger, and an analysis of the kernel file we are loading. If there are zeros at these memory locations in QEMU, that part of your code is broken:

Kernel is loaded correctly? (not 100% sure on this but if it is empty none of your function works)



(see bootloader-stage2.cpp)

Your file (QEMU):



Kernel file (what it’s loading, in little endian)



Testing if your direct block access is working:

Your file (QEMU):

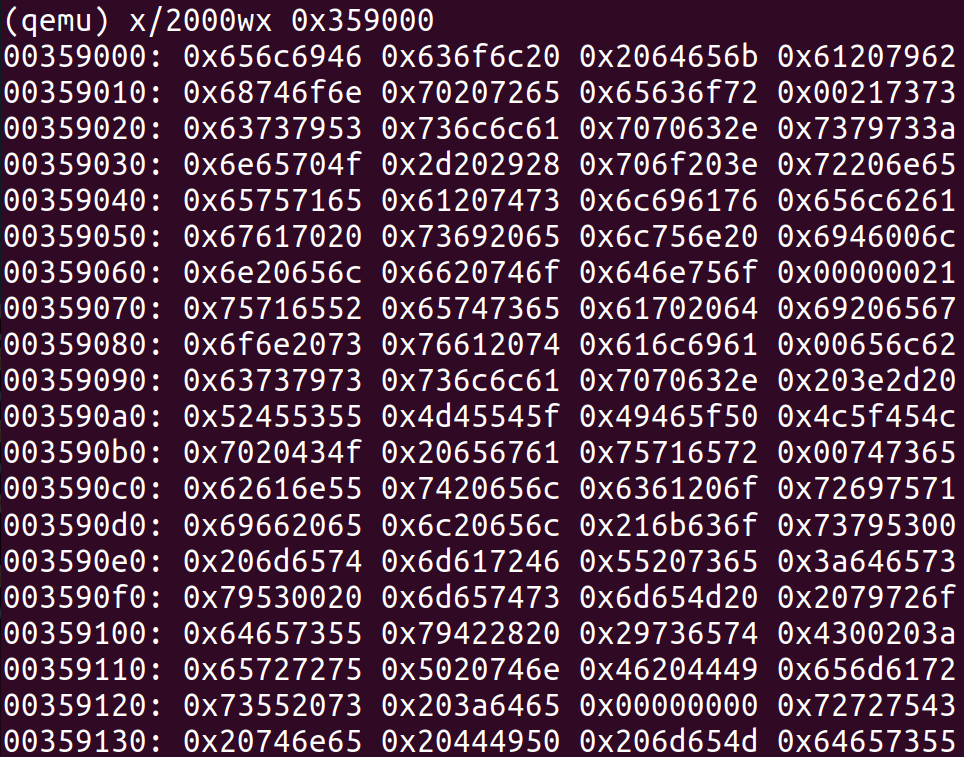


Kernel file (what it’s loading, in little endian)

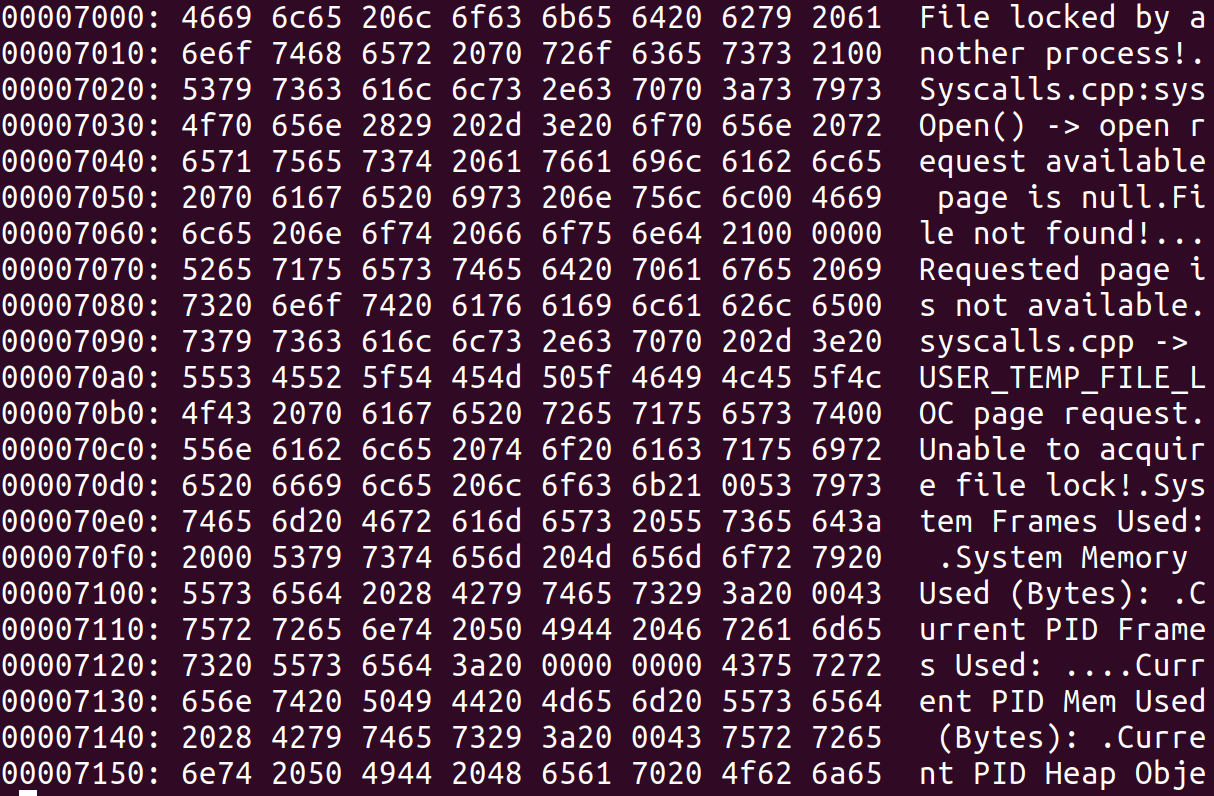


Testing if your indirect block access is working:

Your file (QEMU):



Kernel file (what it’s loading, in little endian)



I don’t know how to test the loadElfFile :(

(It should only be about 5 lines. No if statements or for loops are necessary)