C. (8 point) Repeat assignment 5 part 1, except that now you should NOT use the `fork()` system call but instead create two separate programs, a producer program and a consumer program, where each is then invoked from a separate command shell (you should pass the same parameter *n* to both programs when you invoke them). Make sure you create the variables storing the input parameters *n* in the data section not the stack AND initialize them to a **non-zero** value. The shared memory shall be created using shm_open and both processes shall use a common file name, e.g. /lab7_shm so that both processes can easily find it.

1. Print the start address of the shared buffer form both processes.
    i. Did you expect it to be similar when printed from both processes? and why?
    ii. Was the address printed virtual or physical address?
2. Print the address of n from your running program and also find out where it's stored in the .elf file (executable).
    i. Did the addresses match (printed from the running program vs the one in the program's elf file)? Why?
    ii. Is your program file (i.e. the elf file) a relocatable object module or an absolute module? Which byte in the elf file tells you that?

Hint:

- To get addresses of variables from an elf file (abosolute program, or executable), you need to use:
  objdump --syms lab7 OR
  objdump -D lab7 OR
  readelf -all lab7 OR
  readelf -s lab7

where lab7 is the name of your executable. Note that objdump may not report variables mapped to the .bss section (i.e uninitialized variables → you must make your variable initialized or use readelf).

- Alternatively, you may tell the linker to output a map file using -Xlinker Map=lab7.map in your gcc command line.