Rinad Anwar

OS lab 8

1. **Synchroize.c**

**Output:**



* Shared Memory allows two or more processes to share a given region of memory
* The only trick in using shared memory is synchronizing access to a given region among multiple processes – if the server/producer process is placing data into a shared memory region, the client/consumer process shouldn’t try to access it until the server is done.

**shmget()** is used to obtain a shared memory identifier:

“IPC\_PRIVATE”, which lets the kernel choose a new key – keys are non-negative integer identifiers, their value continually increases to a maximum value, where it then wraps around to zero.

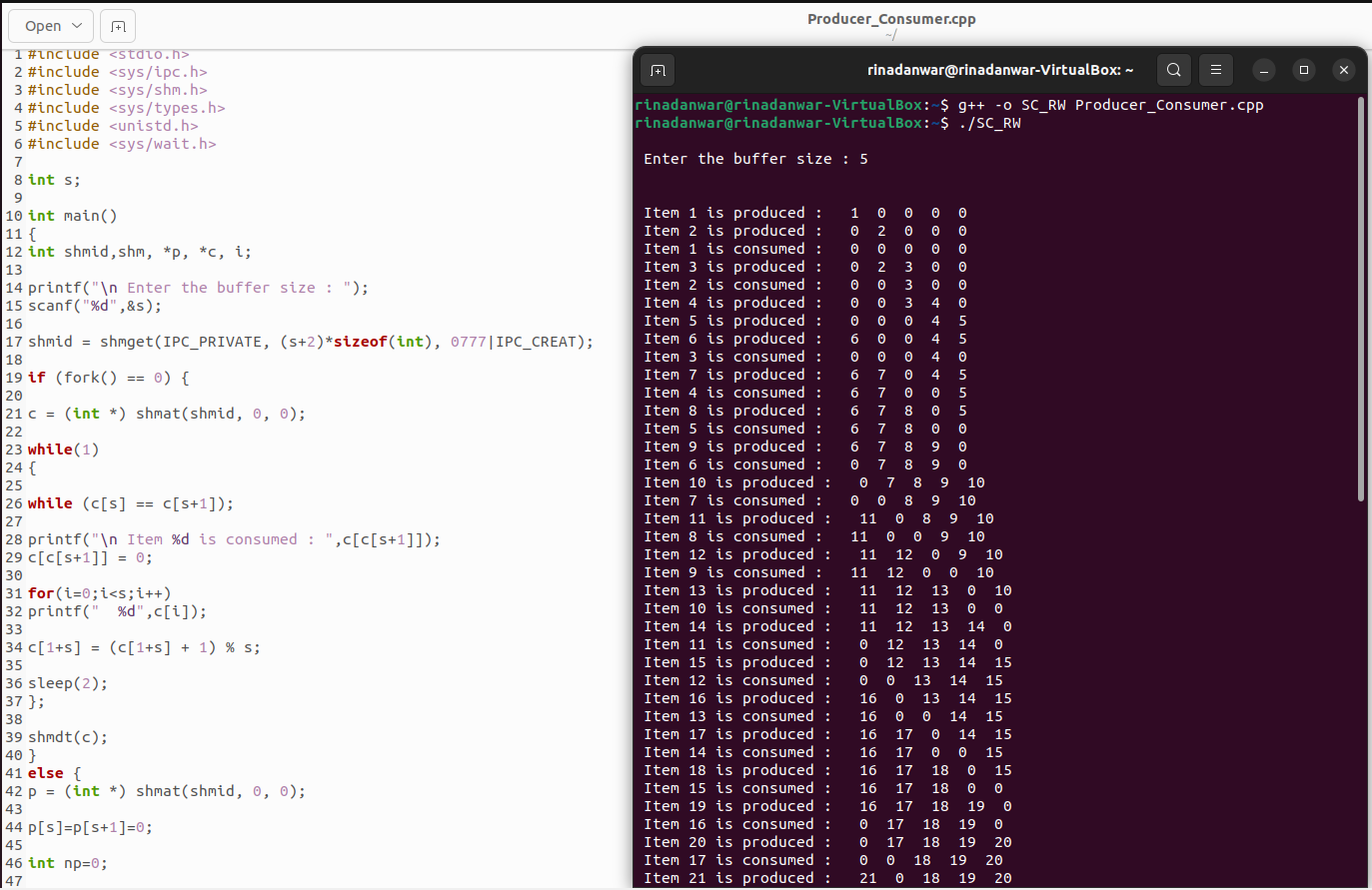
**shmat():** Once a shared memory segment has been created, a process attaches it to its address space by calling shmat(). shmat() returns a pointer to shared memory segment if OK, -1 on error

**shmctl():**  sets up a shared memory segment to remove itself once the process dies

**shmget():** used to create and gain access to an existing segment.

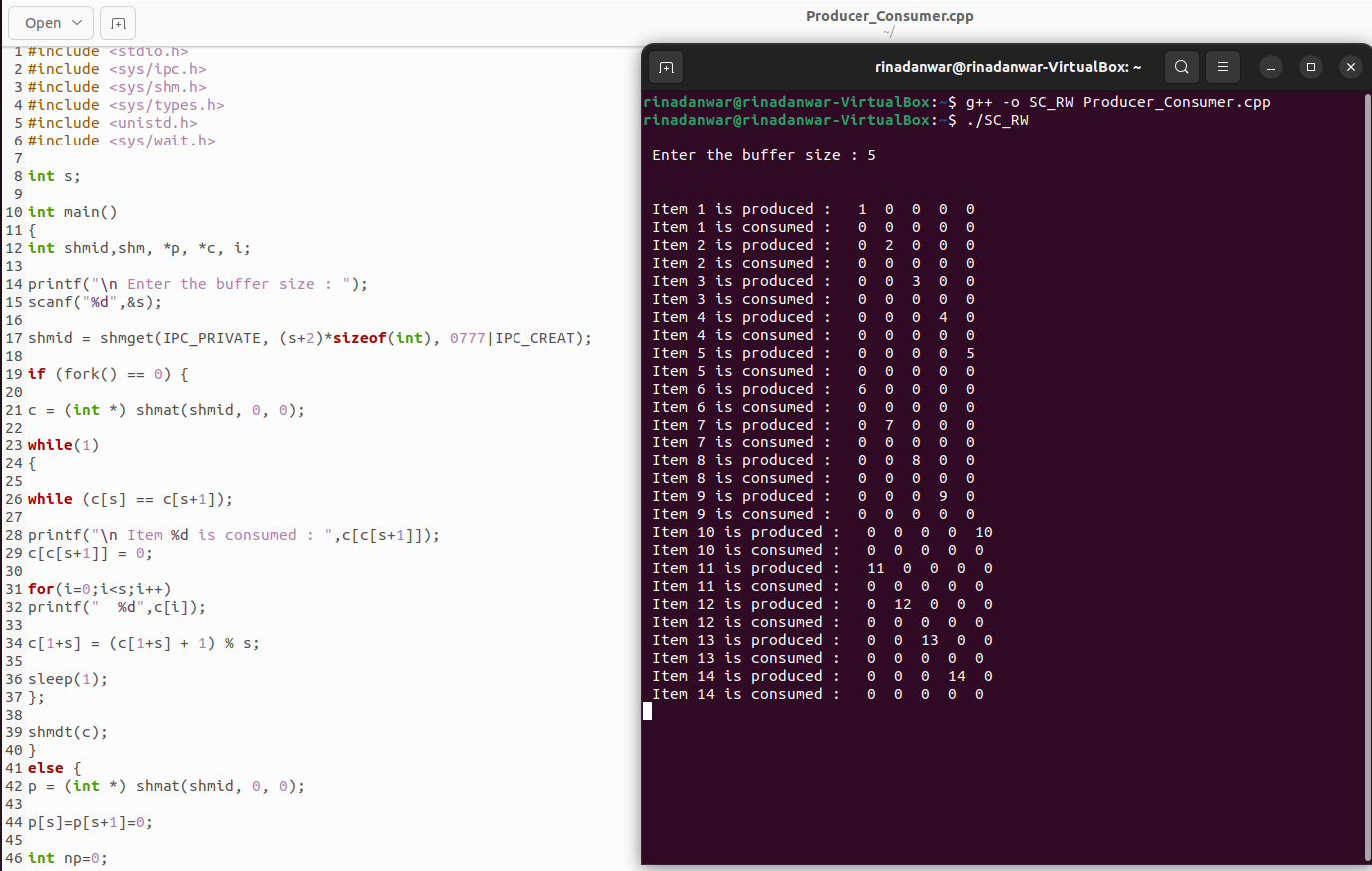
1. **Producer\_Consumer.cpp**

**a) Modify the sleep in the child process to sleep(2). What happens now?**

****

The consumed process sleeps for 2 seconds, as you can see the produced for item 1 is shown in output first, then after 2 seconds consumed process is shown in output. The parent prints even if the child is sleeping.

**b) Modify the sleep in the parent process to sleep(2). What happens now?**

****

Here the child process is not in output unless the parent is printed first, so the system waits for the parent process for 2 seconds and then prints parent and then its child process “consumed”.