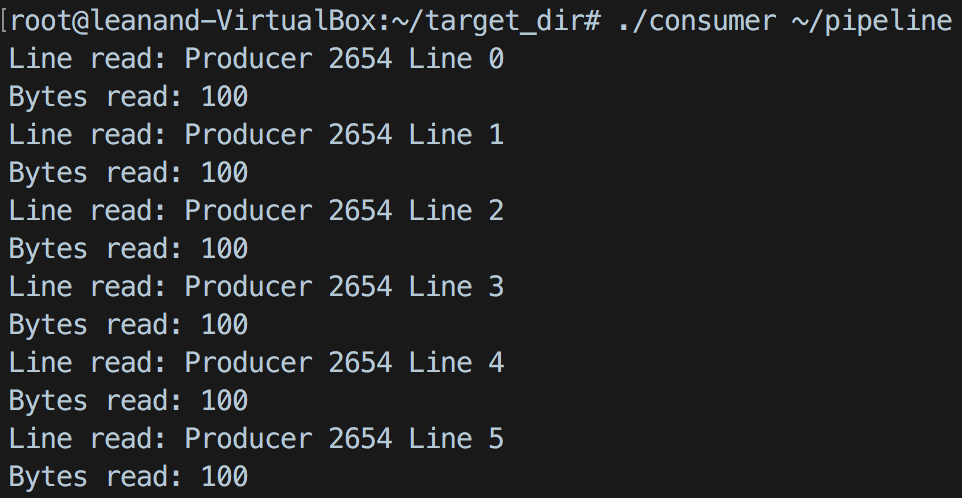
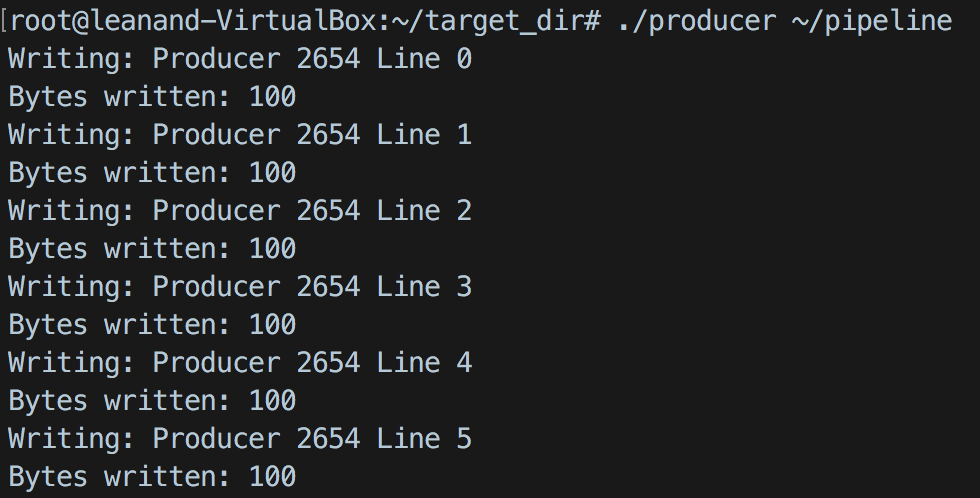
**1. Run one consumer and one producer concurrently.**

When one consumer and producer run concurrently, the producer writes to the pipe, and the consumer reads from the pipe with no errors.

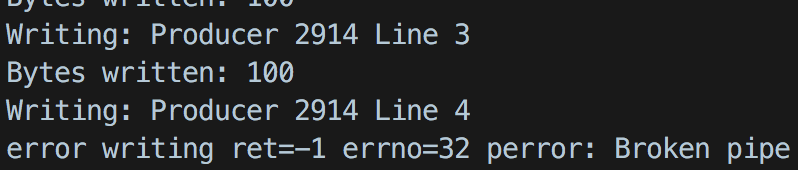
**Consumer**



**Producer**

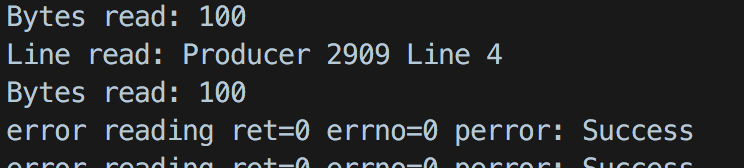


* Kill the producer with Ctrl-C. Leave consumer running. What happens and why?



When consumer is killed, it closes the read end of the pipe. When producer tries to write to closed pipe, it will return error EPIPE. We cannot write to a pipe whose reading end is closed.

* Kill the consumer with Ctrl-C. Leave producer running. What happens and why?

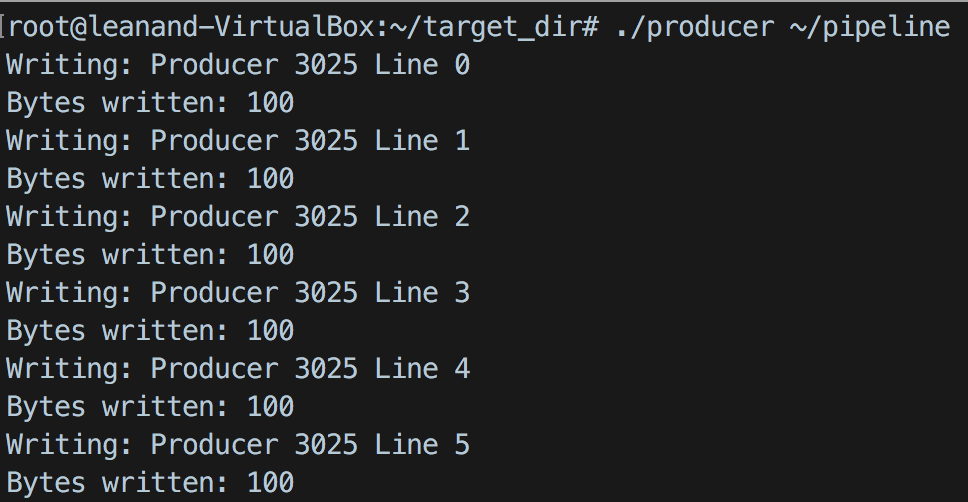


When the producer is killed, it closes the write end of the pipe. So if consumer tries to read a closed pipe, it will return 0 bytes.

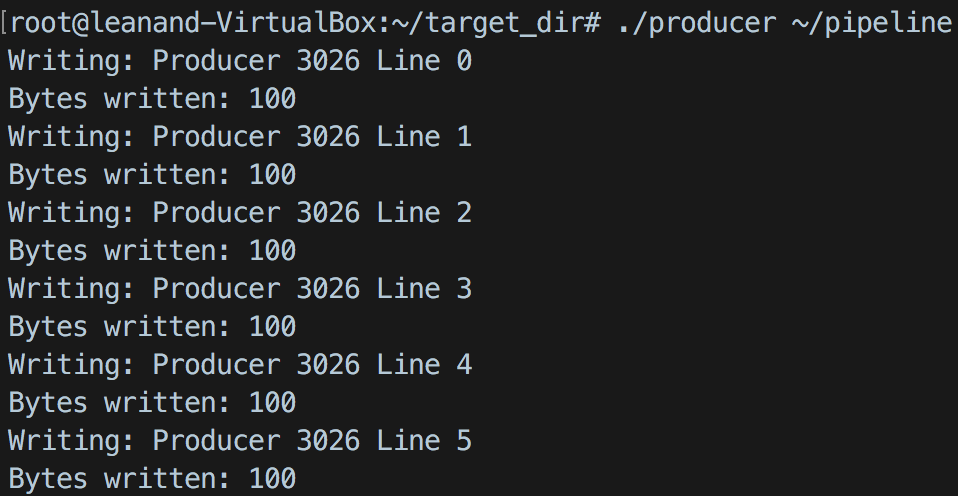
**2. Run one consumer and multiple producers concurrently.**

Multiple producers write line to the pipe and the single consumer reads lines written by both the producers.

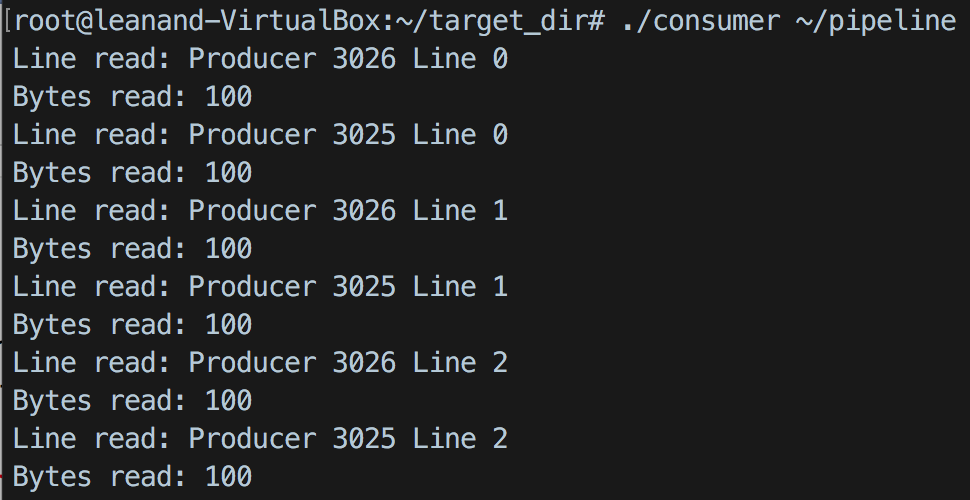
**Producer 1**



**Producer 2**



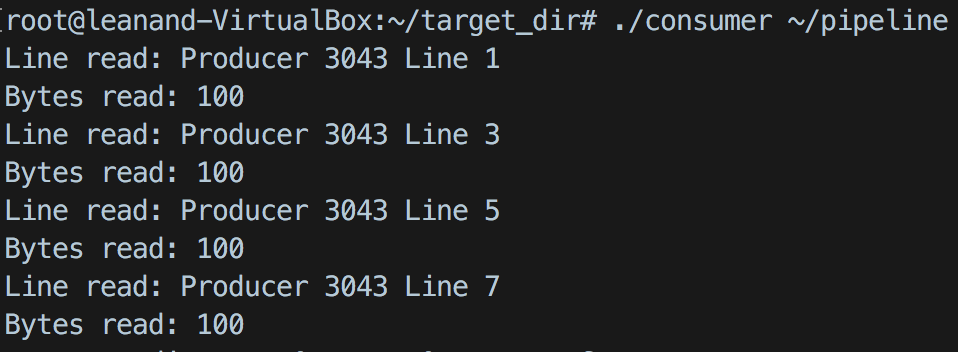
**Consumer**

****

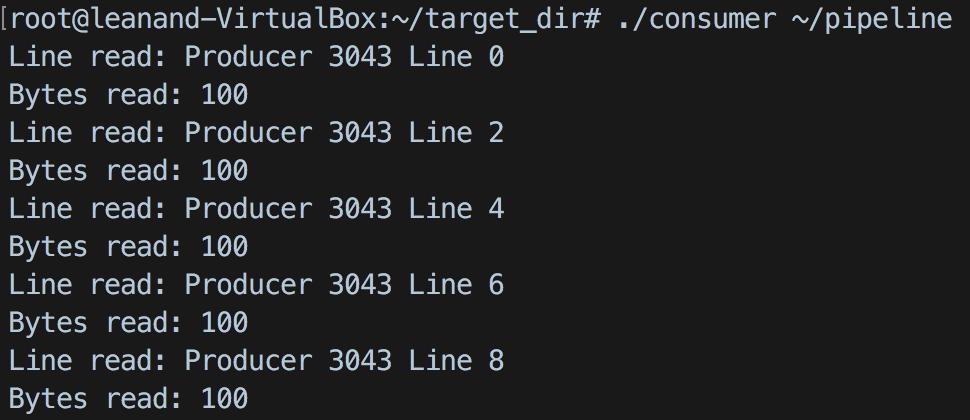
**3. Run multiple consumers and one producer concurrently**

If there are multiple consumer are trying to read from a single producer, the lines gets split between each consumer. In this case, the lines are consumer alternatively. If there are more than two consumers with just one producer, then lines will be read randomly based on lock gained by the consumer.

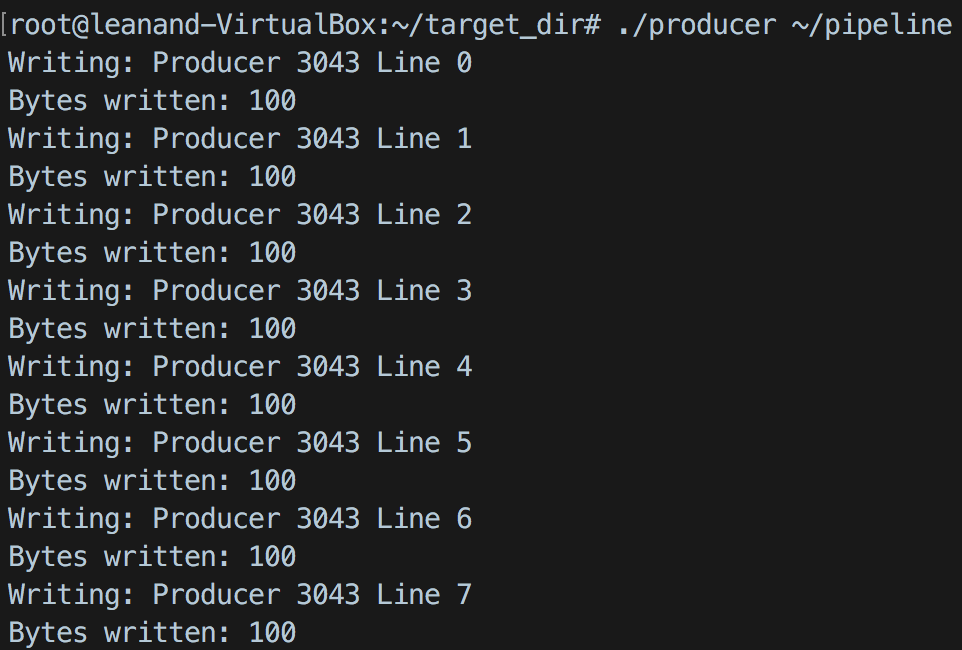
**Consumer 1**

****

**Consumer 2**

****

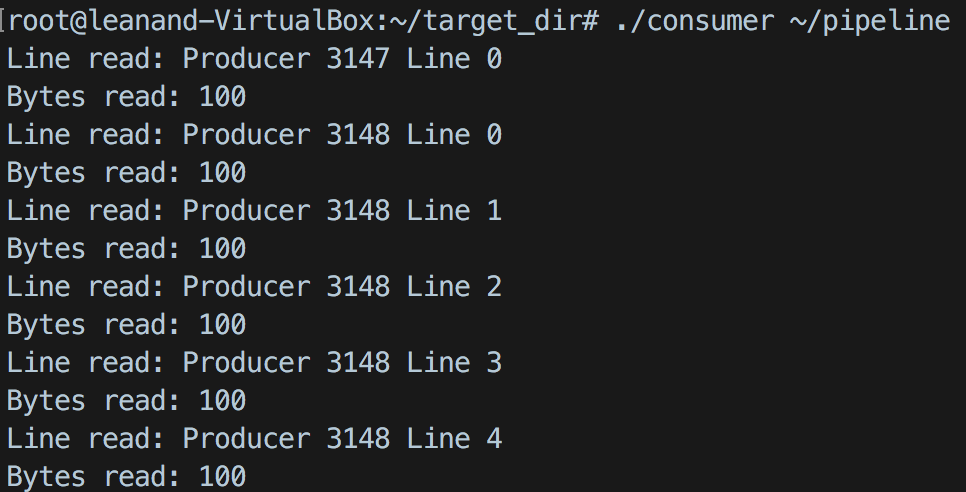
**Producer**

****

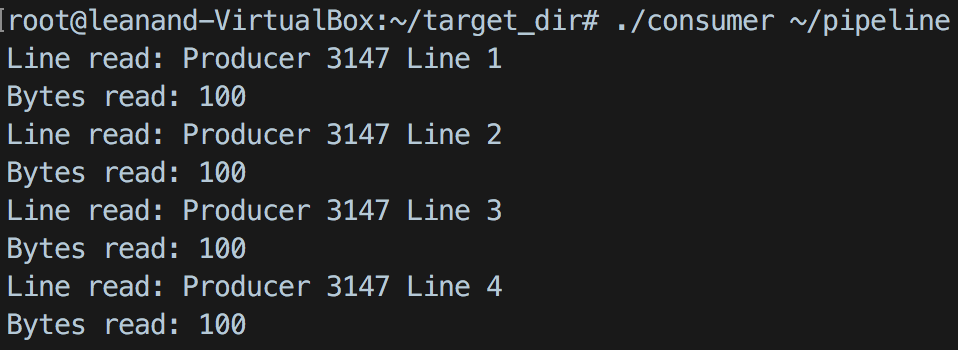
**4. Run multiple consumers and multiple producers concurrently.**

When there are more multiple consumers and multiple producers, the lines are read randomly. In this case, the number of consumers are equal to number of producers. So each line by a producer is read by the same consumer always.

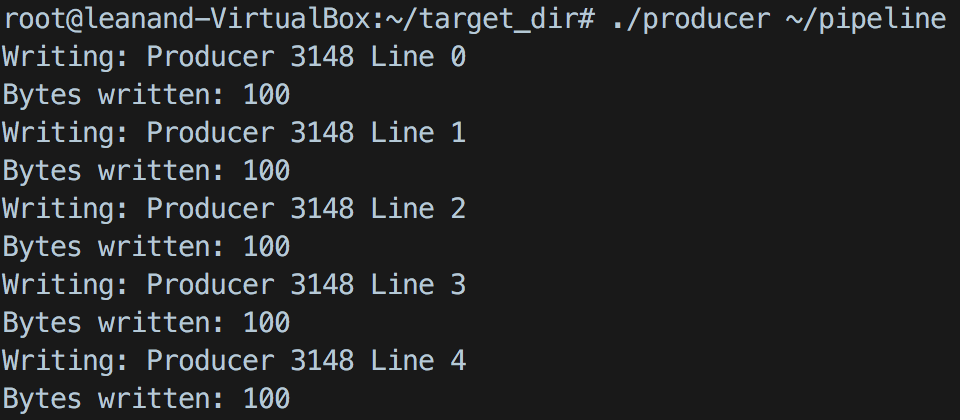
**Consumer 1**

****

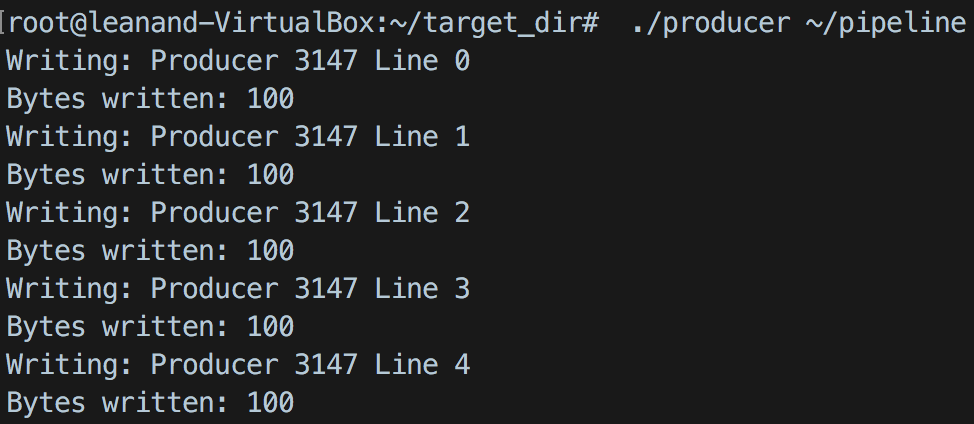
**Consumer 2**

****

**Producer 1**

****

**Producer 2**

****