Shared memory

In this lab, we will look into how processes can share memory using the shmget, shmat system calls.
The input will be entered into the producer's terminal in the following format:

```
3
There are seven
Words in
this input
```

- The first line contains **the number of lines (n)** to be read.
- Each of the n lines to be processed has words containing only English alphabet letters.
- Every line has a newline character at the end.
- Words in a line are separated by **one** space, except for the last word which has a newline after it.
- Every line has at least one word.

There are two processes - the producer and the consumer. The code for both of them should be implemented in separate files.
The producer's task (producer.c):

1. Create a block of shared memory of size 1000 with your selected key.
2. Find the number of lines (n) to read by reading the input from stdin and print it (format given below).
3. Input the next line, read it from stdin, and store it in the created shared memory. Also, print the line (format below). Store the sentence starting from the **second byte**.
4. Store the character `'#'` in the **first byte** to inform the consumer that the shared memory has the line to be processed.
5. Wait until the first byte is marked `'%'` by the consumer, so the producer knows the consumer has processed this line.
6. Repeat steps 3-5 **n times**.
7. After the consumer has consumed all the n lines, the producer stores the character `'$'` in the first byte of the shared memory and exits.

The consumer's task (consumer.c). :

1. Attach to the shared memory block created by the producer using your key.
2. Wait till the first byte has the character `'#'` so the consumer knows a line is ready to be processed.
3. Access the shared memory to count the number of words in the stored line. Words in a line are separated by one space. Print the number of words in that line (format given below)

4. Mark the first byte of the shared memory with the character `'%'` so the producer knows the consumer has processed this line.
5. Repeat steps 2-4 **till the producer stores `'$'` in the first byte.**
6. Print the total number of words in the entire input and exit.

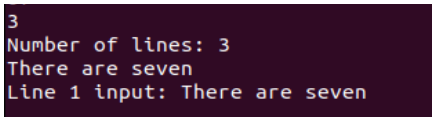
Hints/Note:

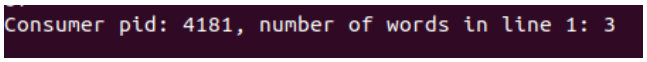
- Use fgets to read a line from stdin with the '\n' at the end.
- Producer clears the shared memory before storing a line
- Make sure both the producer and the consumer detach themselves from the shared memory segment before exiting and the producer marks the shared memory segment for destruction.
- Use atoi() to convert a string to a number, if required.

Execution example: Open two terminal windows. Execute producer.c first in one window and then consumer.c in the second window.

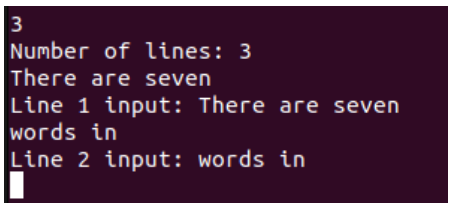
producer.c terminal **consumer.c terminal**

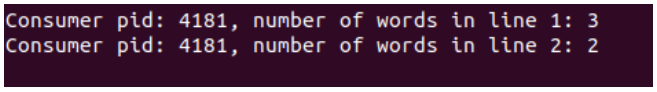
1.) Number of lines and the first line entered as input to the producer



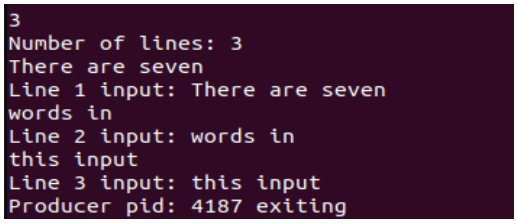


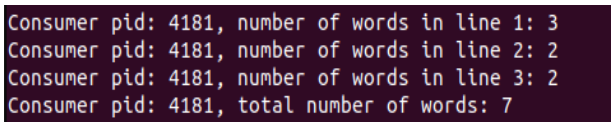
2.) Second line entered as input to the producer





3.) The third line (which is the last line) is entered as input to the producer.





Terminal output examples

Producer.c

```
3
Number of lines: 3
There are seven
Line 1 input: There are seven
Words in
Line 2 input: Words in
this Input
Line 3 input: this Input
Producer pid: 6154 exiting
```

Consumer.c

```
Consumer pid: 6199, number of words in line 1: 3
Consumer pid: 6199, number of words in line 2: 2
Consumer pid: 6199, number of words in line 3: 2
Consumer pid: 6199, total number of words: 7
```

Producer.c

```
5
Number of lines: 5
A B C D
Line 1 input: A B C D
E F G
Line 2 input: E F G
H I J K L M N O
Line 3 input: H I J K L M N O
P Q R S T U
Line 4 input: P Q R S T U
V W X Y Z
Line 5 input: V W X Y Z
Producer pid: 6234 exiting
```

Consumer.c

```
Consumer pid: 6235, number of words in line 1: 4
Consumer pid: 6235, number of words in line 2: 3
Consumer pid: 6235, number of words in line 3: 8
Consumer pid: 6235, number of words in line 4: 6
Consumer pid: 6235, number of words in line 5: 5
```

```
Consumer pid: 6235, total number of words: 26
```