Lab Exercise – 14

AIM :: Implement `Producer-Consumer` problem by using semaphores in Shell Scripting.

Theory ::

Producer-Consumer Problem

The Producer-Consumer problem is a classic example of a multi-process synchronization issue. It involves two types of processes, producers and consumers, that share a common, fixed-size buffer. The producer's job is to produce data and place it in the buffer, while the consumer's job is to remove the data from the buffer. To prevent the consumer from trying to consume from an empty buffer or the producer from trying to add to a full buffer, semaphores are used to control access to the buffer.

Key Points

1. **Buffer Management**: The buffer size is fixed, and synchronization is required to prevent concurrent access issues.
2. **Semaphores**:
   - `sem_empty` keeps track of the empty slots in the buffer.
   - `sem_full` keeps track of the filled slots in the buffer.
   - `mutex` ensures mutual exclusion while accessing shared resources.
3. **Producer Function**:
   - Generates a random item and adds it to the buffer when there’s space.
4. **Consumer Function**:
   - Removes an item from the buffer and processes it when there are items to consume.
5. **Infinite Loop**: Both the producer and consumer run indefinitely for demonstration purposes.

Synchronization Solutions

To address synchronization challenges in the Producer-Consumer problem, mechanisms such as semaphores, mutexes, and condition variables are employed. These tools help:

- Ensure producers wait when the buffer is full.
- Ensure consumers wait when the buffer is empty.
- Prevent race conditions during access to the shared buffer.

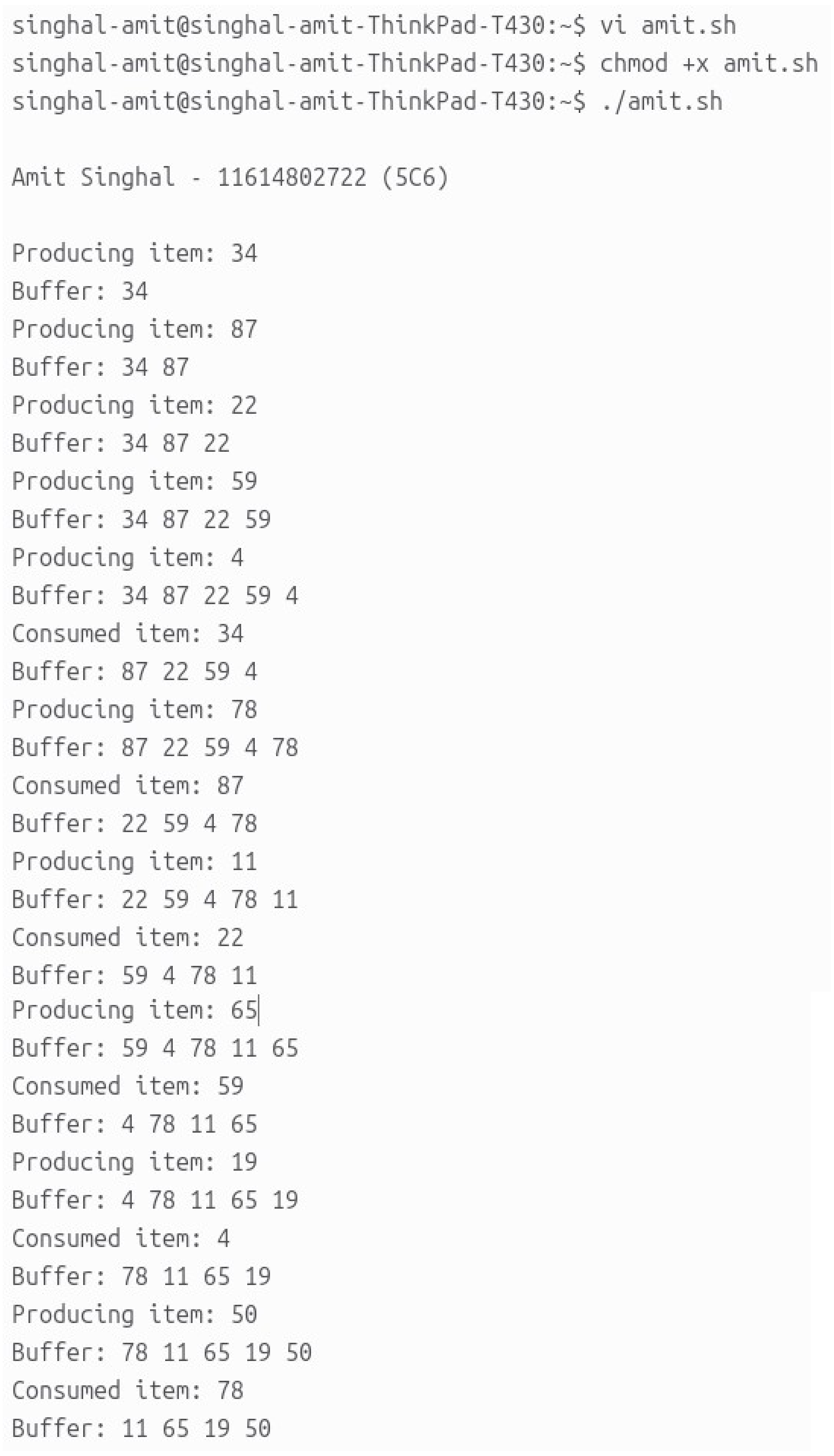
By thoughtfully designing synchronization logic, the Producer-Consumer problem can be effectively managed, facilitating efficient and safe data sharing among concurrent processes.

Source_Code ::

```
echo "Amit Singhal - 11614802722 (5C6)"

# Parameters
buffer_size=5
buffer=()
count=0
sem_empty=$buffer_size
sem_full=0
mutex=1
iterations=20 # Set number of iterations to run

# Functions to produce and consume items
produce_item() { echo $((RANDOM % 100)) }
consume_item() { echo "Consumed item: $1" }

# Producer function
producer() {
    for ((i = 0; i < iterations; i++)); do
        item=$(produce_item)
        echo "Producing item: $item"

        # Wait until there's space in the buffer
        while [[ $sem_empty -eq 0 ]]; do
            sleep 1 # Wait
        done

        # Enter critical section
        ((mutex--))
        buffer+=("$item")
        ((count++))
        ((sem_empty--))
        ((sem_full++))
        echo "Buffer: ${buffer[@]}"
        ((mutex++))
        sleep 1 # Simulate time taken to produce
```

```
  done
}

# Consumer function
consumer() {
  for ((i = 0; i < iterations; i++)); do
    # Wait until there's at least one item to consume
    while [[ $sem_full -eq 0 ]]; do
      sleep 1 # Wait
    done

    # Enter critical section
    ((mutex--))
    item=${buffer[0]}
    buffer=("${buffer[@]:1}") # Remove the first item
    ((count--))
    ((sem_full--))
    ((sem_empty++))
    echo "Buffer: ${buffer[@]}"
    consume_item "$item"
    ((mutex++))
    sleep 1 # Simulate time taken to consume
  done
}

# Start producer and consumer in the background
producer & # Start producer in background
consumer & # Start consumer in background

# Wait for the processes to finish
Wait
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~$ vi amit.sh
singhal-amit@singhal-amit-ThinkPad-T430:~$ chmod +x amit.sh
singhal-amit@singhal-amit-ThinkPad-T430:~$ ./amit.sh

Amit Singhal - 11614802722 (5C6)

Producing item: 34
Buffer: 34
Producing item: 87
Buffer: 34 87
Producing item: 22
Buffer: 34 87 22
Producing item: 59
Buffer: 34 87 22 59
Producing item: 4
Buffer: 34 87 22 59 4
Consumed item: 34
Buffer: 87 22 59 4
Producing item: 78
Buffer: 87 22 59 4 78
Consumed item: 87
Buffer: 22 59 4 78
Producing item: 11
Buffer: 22 59 4 78 11
Consumed item: 22
Buffer: 59 4 78 11
Producing item: 65
Buffer: 59 4 78 11 65
Consumed item: 59
Buffer: 4 78 11 65
Producing item: 19
Buffer: 4 78 11 65 19
Consumed item: 4
Buffer: 78 11 65 19
Producing item: 50
Buffer: 78 11 65 19 50
Consumed item: 78
Buffer: 11 65 19 50
```