Alexandria University
Faculty of Engineering
Comp. & Comm. Engineering
CC373: Operating Systems

جامعة الإسكندرية
كلية الهندسة
برنامج هندسة الحاسب والاتصالات
مادة نظم التشغيل

Lab5
Producer/Consumer Problem

Objectives

- Learn how to use semaphores for mutual exclusion and synchronization.
- Learn how to use shared memory between processes.

Introduction

Each student is required to implement the Bounded-Buffer Producer/Consumer problem as learnt in the lectures. You may have up to 20 producers. Each Producer would declare the live price of a commodity (e.g., GOLD, SILVER, CRUDEOIL, COTTON, …). One Consumer would show a dashboard for the prices of all commodities. Producers and Consumer would run indefinitely sharing the prices through shared memory.

Producers

Each producer is supposed to continuously declare the price of one commodity. For simplicity, we assume that each commodity price follows a normal distribution with parameters (μ, σ^2). Therefore, producers will generate a new random price, share it with the consumer, and then sleep for an interval before producing the next price.

All producers are processes running the same codebase. Producers are to be run concurrently, either in separate terminals, or in the background. While running a producer, you will specify the following command line arguments:

- Commodity name (e.g., GOLD – Assume no more than 10 characters.)
- Commodity Price Mean; μ – a double value.
- Commodity Price Standard Deviation; σ – a double value.
- Length of the sleep interval in milliseconds; T – an integer.
- Bounded-Buffer Size (number of entries); N – an integer.

Therefore, the command "./producer NATURALGAS 7.1 0.5 200 40" would run a producer that declares the current price of Natural Gas every 200ms according to a normal distribution with parameters (mean=0.5 and variance=0.25). The size of the shared bounded buffer is 40 entries.

Let the producer log what he does by writing to `stderr`. For instance, a producer may write:

```
[12/31/2022 21:45:20.356] GOLD: generating a new value 1820
[12/31/2022 21:45:20.822] GOLD: trying to get mutex on shared buffer
[12/31/2022 21:45:21.128] GOLD: placing 1820 on shared buffer
[12/31/2022 21:45:22.596] GOLD: sleeping for 200 ms
[12/31/2022 21:45:22.635] GOLD: generating a new value 1830
[12/31/2022 21:45:22.984] GOLD: trying to get mutex on shared buffer
[12/31/2022 21:45:25.353] GOLD: placing 1830 on shared buffer
[12/31/2022 21:45:26.142] GOLD: sleeping for 200 ms
```

You may get the current time in nanosec resolution through the `clock_gettime()` system call.

Consumer

The consumer is to print the current price of each commodity, along the average of the current and past 4 readings. An Up/Down arrow to show whether the current Price (AvgPrice) got increased or decreased from the prior one. Until you receive current prices, use `0.00` for the current price of any commodity.
For simplicity, let's limit the commodities to GOLD, SILVER, CRUDEOIL, NATURALGAS, ALUMINIUM, COPPER, NICKEL, LEAD, ZINC, MENTHAOIL, and COTTON. Show the commodities in alphabetical order.
While running the consumer, you will specify the following command line argument:

- Bounded-Buffer Size (number of entries); N – an integer.

Below is an example output.

```
+-------------------------------------+
| Currency      |  Price    | AvgPrice |
+-------------------------------------+
| ALUMINIUM     |    0.00   |    0.00  |
| COPPER        |    0.00   |    0.00  |
|                    ...              |
| GOLD          | 1810.31↑  | 1815.25↑ |
|                    ...              |
| SILVER        |   22.36↓  |   22.80↓ |
| ZINC          |    0.00   |    0.00  |
+-------------------------------------+
```

Required System Calls

You are to use System V IPC for interprocess communication. It provides facility for

- Shared Memory; see this example.
- Semaphores; see this example.

Hints

- You may use the standard C++ Random library to generate normally distributed random variables. An example can be found in its reference.
- You may use escape sequences to clear the screen and place the cursor at the top of the screen. It is platform independent.

```
printf("\e[1;1H\e[2J");
```

- You may use escape sequences to move the cursor to a certain position. For instance, the following statement would move the cursor to line 5 and column 10.

```
printf("\033[5;10H");
```

- You may use Linux terminal commands to print color text on the terminal. See this example.
- Use the format specifier `"7.2lf"` while printing prices.

Deliverables

- The directory containing your code should have three files:
  - `producer.cpp` file whose main function accepts the arguments as described above.
  - `consumer.cpp` file whose main function accepts the arguments as described above.
  - `Makefile` that will build the two files and create two output files **`producer`** and **`consumer`**.

- Step outside the directory containing your code.
- Name your work directory as yourid-lab# (for id: 5678 and lab#5, the directory should be called 5678-lab5)
- Create a .tar.gz file using

```
tar cvfz 5678-lab5.tar.gz <path_of_directory>
```

- Append .pdf to the filename to be able to upload your project