Shared Memory | SCOMP-PL3

1. Implement a solution that sends the number and name of a student between two processes not related hierarchically, a writer and a reader.
   - The writer must create a shared memory area, read the data from the keyboard and write them in the shared memory.
   - The reader should read the data from the shared memory and print them on the screen.

   Note: the writer must be executed before the reader.

2. Implement a solution that sends the product code, description, and price between two processes not related hierarchically, a writer and a reader.
   - The writer must create a shared memory area, read the data from the keyboard and write them in the shared memory.
   - The reader should read the data from the shared memory and print them on the screen.

   Note: the writer must be executed before the reader.

3. Implement a solution that tests the speed of two transfer methods between two processes – pipes and shared memory.
   - Start by filling an array of 100 000 structures with an integer and the "ISEP – SCOMP 2020" string.
   - Start a timer and copy that array between the two processes using shared memory. Start a timer and use a pipe to transfer the same amount of data between processes.

4. Implement a solution that allows you to share an array of 100 characters between two processes not related hierarchically, a writer and a reader.
   - The writer must create a shared memory area, generate 100 random chars (between 'A' and 'Z') and write them in the shared memory.
   - The reader should read the 100 values, calculate the sum of ASCII code and print all the chars and the average of the 100 ASCII codes.

   Note: the writer must be executed before the reader

5. Implement a program that creates a shared memory area to store two integers and initializes those integers with the values 8000 and 200 and creates a new process. Parent and child must perform the following operations 1.000.000 times:
   - The father will increase the first value and decrease the second value.
   - The child will decrease the first value and increase the second value.

   Only write the value on the screen at the end. Review the results. Will these results always be correct?

6. Implement a solution that tests the speed of two transfer methods between two processes – pipes and shared memory. Start by filling a 1.000.000 array with numbers. Start a timer and copy that array between the two processes one element at a time using shared memory. Start a timer and use a pipe to transfer the same amount of data between processes. Test for different sizes of data.

7. Implement a solution that allows you to share an array of 10 integers between two processes not related hierarchically, a writer and a reader.
   - The writer must create a shared memory area, generate 10 random numbers between 1 and 20 and write them in the shared memory.
   - The reader should read the 10 values, calculate and print the average.

   Note: the writer must be executed before the reader.

8. Implement a program that creates a shared memory area to store an integer, initializes this value to 100, and creates a new process. Parent and child must perform the following operations 1.000.000 times:
   - Increase the value;
   - Decrease the value;
   - Only write the value on the screen at the end.

   Review the results. Will these results always be correct?

9. Implement a program to determine the biggest element of an array in a parallel/concurrent environment. The parent process should:
   - Create an array of 1000 integers, initializing it with random values between 0 and 1000;
   - Create a shared memory area to store an array of 10 integers, each containing the local maximum of 100 values;
   - Create 10 new processes;
   - Wait until the 10 child processes finish the search for the local maximum;
   - Determine the global maximum;
   - Eliminate the shared memory area.

   Each child process should:
   - Calculate the largest element in the 100 positions;
   - Write the value found in the position corresponding to the order number (0-9) in the array of local maximum.

10. Using the program written for question 9, determine which is the number of processes which gives the best performance in your systems. Discuss the results taking into account the architecture of your computer.

11. Implement a program similar to 9, but now the child processes send the local maximum using a pipe, shared by all processes.

12. Implement a program that allows the exchange of data concerning a student between two processes (number, name and grades of a set of classes). The data to be exchanged are represented in the following struct aluno.

```
1  #define STR_SIZE 50
2  #define NR_DISC 10
3  struct aluno{
4      int numero;
5      char nome[STR_SIZE];
6      int disciplinas[NR_DISC];
7  };
```

   The parent process should:
   - Create a shared memory area for data exchange. Check the need to add one or more variables to synchronize the writing and reading of data operations;
   - Create a new process;
   - Fill the shared memory area in accordance with user-entered information;
   - Wait until the child process ends.
   - Eliminate the shared memory area.

   The child process should:
   - Wait for the student data;

- Calculates the highest, the lowest and the average grade;
- Print the information on the screen.

13. Implement a program that allows to optimize the search of words in a set of text files in parallel/concurrent environment.
    The parent process should:
    - Create an area of shared memory. The memory area must contain, for each child process, the following information:
      – path to the file;
      – word to search;
      – an integer to store the number of occurrences.
    - Create 10 new processes;
    - Fill the shared memory area with the information for each child process;
    - Wait until the child processes finish their search;
    - Print the number of occurrences determined by each child;
    - Eliminate the shared memory area.

    Each child process should:
    - Open the text file assigned to it by the parent (can/should be different for each child process);
    - Determine the number of occurrences of the word to search;
    - Write the number of occurrences in their position in the shared memory area.

14. Implement a program that creates a new process. One will be the producer and the other the consumer. Among them should be created a circular buffer to store 10 integers and the necessary synchronization variables in a shared memory area. The producer puts increasing values in the buffer that should be printed by the consumer. Consider that 30 values are exchanged between them.

15. Solve again question 14, but now the synchronization between the processes should be supported by pipes, instead of shared variables.