Communicating with processes through pipes | SCOMP-PL2

1. Write a program that creates a child process and establishes with it a communication channel through a pipe. The parent process starts by printing its PID and then sends it to its child through the pipe. The child process should read the parent´s PID from the pipe and print it.

2. Write a program that creates a child process and establishes with it a communication channel through a pipe. The parent process reads an integer and a string from *stdin* and sends both through the pipe. The child process should read those values from the pipe and print them.

    a. Solve the exercise by first sending an integer and then a string in two separate write operations;

    b. Solve the exercise by using a structure to send both values in one write operation.

3. Write a program that creates a child process and establishes with it a communication channel through a pipe. The parent process should send two messages, “Hello World” and “Goodbye!”, and then closes the communication channel. The child process should wait for data and print it as soon as it is available, terminating after all data is received. The parent process must wait for its child to end, printing its PID and exit value.

4. Write a program that creates a child process and establishes with it a communicating channel through a pipe. The parent process should send the contents of a text file to its child through the pipe. The child should print all the received data. The parent must wait for its child to end.

5. Two processes, client (the child process) and server (the parent process), communicate via two pipes “Up” and “Down”.

    Server (parent process)
    ^ Up |
    | v Down
    Client (child process)

    The client reads a message from *stdin* and sends it to the server via the Up pipe, then waits for the server's answer on the Down pipe. The server is specialized in converting characters from lower case to upper case and vice-versa. Therefore, if the client sends the message "lower case" via the Up pipe, the server will read the message, convert it, and send "LOWER CASE" via the Down pipe. When the client receives the message from the server, it prints it out. You may assume the maximum size of any message is 256 bytes.

6. Given two integer arrays vec1 and vec2, with 1000 elements each, write a program that creates 5 child processes that communicate with the parent through a pipe, to concurrently sum the two arrays. Each child should calculate tmp += vec1[i] + vec2[i] on 200 elements, then sending tmp to its parent through the pipe. Ensure that each child computes different positions of the arrays. The parent process should wait for the 5 partial sums and then calculate the final result. **There must be only one pipe, shared by all processes.**

    a. Explain why the there is no need for a synchronization mechanism? Take a look at the pipe function complete man pages.

Communicating with processes through pipes | SCOMP-PL2

7. Given two integer arrays vec1 and vec2, with 1000 elements each, write a program that creates 5 child processes to concurrently sum the two arrays. Each child should calculate vec1[i] + vec2[i] on 200 elements, sending all those values to its parent. Ensure that each child computes different positions of the arrays. The parent process should wait for all the 1000 values and then store them in a result array, in the correct position. **Use 5 pipes, one for each child.**

8. Write a program that creates 10 child processes to play the game "Win the pipe!". There must be only one pipe, shared by all processes. The game's rules are as follow:

a. The parent process fills, each 2 seconds, a structure with a message "Win" and the round's number (1 to 10) and writes this data in the pipe;

b. Each of child processes is trying to read data from the pipe. The child that succeeds should print the winning message and the round's number, ending its execution with an exit value equal to the winning round's number;

c. The remaining child processes continue to try to read data from the pipe;

d. After all child processes terminate, the parent process should print the PID and the winning round of each child.

9. A retail company stores the sales of the previous month in a *sales* array and wants to know which products have sold more than 20 units in a single sale. As the array size is quite large (50 000 elements) a concurrent solution to the problem is necessary. Write a program that creates 10 child processes, with each one responsible for searching 5000 sale records. Consider that each sale record is as follows:

```
typedef struct{
  int customer_code;
  int product_code;
  int quantity;
}
```

Whenever a child finds a product that meets the search criteria it should send the product's code to the parent process. The parent process should record those codes in the array *products*, printing it whenever all child processes have been terminated. The values on the sales array should be filled with random values.

10. Write a program that simulates a betting system. Assume that the child process starts the game with a credit of 20 euros. The game has the following rules:

a. The parent process generates a random number between 1 and 5;

b. Then, it writes 1 in the pipe, notifying the child that it can make another bet, or 0, if the child's credit ended;

c. The child process reads the number sent by the parent process and makes a bet or terminates, accordingly. To make a bet, the child should generate a random number between 1 and 5 and send it to the parent process;

d. The parent process waits for the child's bet or by its termination, accordingly. If the child's bet is right, its credit grows 10 euros. If the bet is wrong, its credit diminishes 5 euros;

e. The parent process sends the new credit's value to the child process. The child process prints it.

Communicating with processes through pipes | SCOMP-PL2

11. Write a program that creates 5 child processes. Connect all 6 processes with a "ring" topology through pipes: the parent process will be connected to child 1, child 1 is connected to child 2, …, and child 5 is connected to the parent process. The goal is to find the greatest random number generated by all processes:

a. Each process generates a random number between 1 and 500. Then, it prints it along with its PID;

b. Then, the parent process sends its generated number to child 1;

c. Child 1 compares the parent's number with its own random number and sends the greater of the two to child 2;

d. All other processes follow the same behavior, until child 5 sends the greater number to the parent process;

e. The parent process prints the greatest random number.

12. Consider a supermarket with 5 barcode readers distributed along the shop. Every time a customer uses a barcode reader the product's name and its price should be printed on the screen. Simulate this system through processes and pipes:

a. The parent process has access to the products database;

b. Each child process represents a barcode reader;

c. There is a pipe shared by all 5 child processes to request the product's information;

d. The parent process replies to the requesting child with the corresponding product's information through a pipe that it shares only with it.

13. Consider a factory line composed of 4 machines and a storage area

a. Machine M1 cuts 5 pieces, 5 pieces at a time, transfers the pieces to M2 and notifies M2;

b. Machine M2, folds the pieces, 5 at a time, transfers the pieces to M3 and notifies M3;

c. Machine M3 welds the pieces, 10 pieces at a time, transfers the pieces to machine M4 and notifies M4;

d. Machine M4 packs 100 pieces at a time, transfer them to storage A1 and adds the produced parts to the inventory.

Write a program which uses processes to represent machines, pipes to represent the flux of pieces. The program should simulate the production of 1000 pieces. All operations must be correctly printed on screen.