PPC project

Cambiecolo

Razmig Kéchichian, Stéphane Frénot, Frédéric Le Mouël and Romain Pujol

1. Goal

The goal of this programming project is to design and implement a multi-process and multi-thread multi-player game in Python. This document defines the functionalities that it has to implement at a minimum. Some questions are deliberately left open for you to propose your own solution. Any extensions and developments will act in your favor in the evaluation. Be rigorous and creative!

2. Presentation

2.1. Minimum specifications

Cambiecolo is the environmentalist cousin of the Cambio card game. Its goal is presenting a hand of 5 cards of the same transport means. The player who succeeds is awarded the points of the transport they put together. The game deals as many different types of transports as there are players. Possible transport means are: airplane, car, train, bike and shoes. Each player receives 5 random cards, face down, e.g. if there are 3 players, 15 cards of 3 transport means are distributed, 5 cards per transport. A bell is placed in the middle of the players. As soon as cards are distributed, players start exchanging by announcing the number of cards they offer, from 1 to 3 identical cards, without showing them. They exchange the same number of cards with the first player to accept the offer. This continues until one of the players rings the bell and presents a hand of 5 identical cards, scoring the points of the transport they have grouped.

2.2. A minimal design

Cambiecolo involves 2 types of processes:

- `game`: implements the game session, keeping track of current offers and the bell.
- `player`: interacts with the user [1], the `game` process and other `player` processes keeping track of and displaying the hand and current offers. Interactions with the `game` process are carried out in a separate thread.

Inter-process communication: Current offers are stored in a shared memory accessible to `player` processes to update their own offer as well as to check current offers. Communication between `player` processes takes place in a message queue requiring a carefully designed exchange protocol.

3. Implementation

3.1. Design

Start with a diagram to better visualize the interactions between processes/threads. State machine diagrams can be very helpful during this stage. The following points need to be addressed and justified:

- relationships between processes (parent-child or unrelated)
- messages exchanged between processes along with their types
- data structures stored in shared memory and how they are accessed
- synchronization primitives to protect access to shared resources or to count resources
- signals exchanged between processes, if any, and their types
- tubes involved in pairwise process communication, if any, and their types

⟹ Write down a Python-like pseudo-code of main algorithms and data structures for each process/thread.

3.2. Implementation plan

Plan your implementation and test steps. We strongly encourage you to first implement and test every process/thread separately on hard-coded data in the beginning. Then you can implement and test each pair of communicating processes/threads, again on hard-coded data in the beginning, if necessary. Only after implementing and testing each inter-process communication you can put all processes together and test the game. Think about how to automate the startup and the proper shutdown of the game, freeing all resources. Identify possible failure scenarios and think about recovery or termination strategies.

---

1 If necessary, refer to this Stackoverflow thread for non-blocking console input solutions in Python.

4. Deadlines and deliverables

3/12/2021	project proposal is published on Moodle
17/12/2021 at 18:00 the latest	students deposit project preparation in the mailbox of the teacher in charge of their group, document is in paper format, it details the design and the implementation plan
3/1/2022 noon the latest	commented documents are available to students at the secretary's office
TP3 session	20-minute max demonstration per project with the teacher in charge of your group
21/1/2022 midnight	submission of project code archives on Moodle