PPC project

Freak out!

Razmig Kéchichian and Guillaume Bono

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. Any extensions and developments will act in your favor in the evaluation. In addition, some points are deliberately left open for you to propose your own solution. Be rigorous and creative!

2. Presentation

2.1. Minimum specifications

The program implements a game for 2 players at least. The game requires high reactivity on the part of players: they do not take turns to play, rather, the player who decides to play proceeds immediately. The winner is the player who first empties her hand of cards, picked from a pile. The pile contains two types of cards, red and blue, numbered 1 to 10. The game starts with a random card on the board, and each player is given a hand of 5 cards, all picked randomly from the pile. Players can only play cards having values immediately smaller or greater than a card already on the board, or the card of the other color having the same value. For example, if there's the red card 8 on the board, players can only play red card 7, red card 9 or blue card 8. A single card is played at a time. Playing a wrong card is sanctioned: the player keeps his card and is given another one from the pile. Taking too long to play, whether the player can actually proceed or not, is sanctioned by giving the player a card from the pile. If the pile is emptied while players still hold their hands, everyone loses!

2.2. A minimal design

The game involves 2 types of processes.

- `board`: Keeps track of, updates and communicates the state of the game, validates players' moves and informs them accordingly.
- `player`: Interacts with the player reading her moves, keeps track of and displays her hand, displays the board. Display operations are carried out in a separate thread.

Inter-process communication: Communication between processes `board` and `player` takes place in a message queue. Process `board` waits for a move, upon receiving one, validates it updating the state of the game if the move is valid. Either way, process `board` informs the player, purging any moves it received in the meantime from the other player, informing her accordingly. Process `player` waits for input with a time-out of predefined duration[1], by the end of which if no input is read, a card is picked from the pile, otherwise the input move is communicated to process `board` for validation. If the move is invalid, a card is picked from the pile, either way the hand is updated. The pile containing the cards is stored in a shared memory accessible to both `board` and `player` processes.

3. Solution

3.1. Design

Start with a diagram to better visualize the interactions between processes. State machine diagrams can be very helpful during the design 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.

1 Refer to this Stackoverflow thread for non-blocking console input solutions in Python

3.2. Implementation plan

Plan your implementation and test steps. We strongly encourage you to first implement and test every process separately on hard-coded data in the beginning. Then you can implement and test each pair of communicating processes, again on hard-coded data in the beginning, if necessary. Only after implementing and testing each inter-process communication you can put 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.

4. Deadlines and deliverables

9/12/2019	project proposal is published on Moodle
20/12/2019 at 17:00 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
6/1/2020 noon the latest	corrected 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
25/1/2020 midnight	submission of project code archives on Moodle