PROJECT

The economic situation is difficult in this period of health uncertainties. Restaurant owners are particularly put to the test by having to comply with strict guidelines: reduction of the tonnage and conservation contact details of customers (guests) to identify contact cases. After we get out of lockdown, the aim of this lab is to help restaurant owners comply with sanitary rules .
To do this, we consider that a restaurant has a fixed number of tables, each having a number maximum places and can accommodate a group. For example, a restaurant might have **3 tables of 2 seats, one table of 4 and a table of 6 seats**. The guests arrive one after the other, announcing themselves (for tracing contact cases); **the first guest in a group must announce the expected number** of people in the group, the following just indicate the first person in the group. When a new group is announced, the restaurateur allocates the smallest available table that can accommodate the group.
At any time, there can be control by the authorities. As the amount of the fine is high, the restaurateur must show that he complies with the rules: the authorities check the allocation of tables as well as the keeping of the "Reminder book" in which the guests and their group are noted.

A table can only accommodate one group at a time, but can of course be used several times: when a group has finished its meal, the table becomes available again and the restaurant owner can assign it to a new group.

Work to be done

We ask to realize the following programs in C language:
1. The **restaurant** program is active for the duration of the service. The first argument is the duration of a meal (expressed in milliseconds to speed up testing and debugging sessions) and the following are the capacities of the tables. At the end of the service, the restaurateur displays the total number of guests and groups welcomed.
2. The program **convive (**that means guest) simulates a guest. The first argument is the name of the guest. The second argument has a different meaning for the first guest in a group or for the following:
- if the guest is the first in a group, he must state as the second argument the number of people expected for the group (including himself). If there is not a sufficient table for the group, the restaurateur immediately informs him and the guest ends.
- the following guests must announce as the second argument the name of the first guest of the group. If this is not present, the guest ends immediately as before.

The guest remains active throughout the meal, which cannot begin until the last member of the group has arrived or curfew time is approaching. To simplify the implementation, the end of meals is indicated by the restaurant owner. When this signals the end of the meal, the guests can end.
3. The program **police** simulates a control of the authorities. It must display the names of the guests currently present at each table as well as the current status of the reminder book, which lists all of the groups that have been served as well as those currently seated.
4. The program **fermeture** (that means closure) indicates that the curfew time is approaching: the restaurant owner must then refuse access to any new guest and start meals at incomplete tables. The program **fermeture** only indicates a situation to the restaurant owner and does not wait for the actual closure: groups already installed (even if all members have not yet arrived) can still finish their meal. If a new member of an installed group arrives after closure, they must be repressed.
When the last table has finished its meal, the restaurant program can then end and display the number of guests as well as the number of groups served.

Obviously, you will avoid any active waiting, even slowed down. To share information between multiple processes, you will only use POSIX shared memory, excluding any other mechanism such as file, pipe, etc. Likewise, for synchronization, you will only use POSIX semaphores, at the exclusion of any other mechanism such as barrier, signal, lock, etc.
You will write a report including in particular the description of the synchronizations (characteristic event of the synchronization, actors concerned, mechanism used) as well as the justification of the information placed in shared memory.

IMPLEMENTATION

The return code for your programs should indicate whether an error has occurred or not. In case of error, you adopt the simple strategy of terminating the execution of the affected program with an explicit message. The dismissal of a guest (no table, announced closure, etc.) should not be considered an error.
To simplify the implementation, we will assume that the following conditions are true. Unless mentioned otherwise, you are not asked to verify them yourself.
- There is only one restaurant in the system.
- The names of the guests are limited to 10 characters (limit to be checked).
- The names of the guests can contain any character, but must start with a character different from a number (to distinguish the second argument **convive**)
- The name of a guest is enough to identify him, in other words the name is unique.
- The maximum capacity of a table is 6 guests (limit to be checked).
- The number of tables in the room and the number of groups welcomed during the opening of the restaurant are not limited.

To manage the duration of meals, it is suggested to use the **sem_timedwait** function and to carefully read the manual for the meaning of the parameter indicating the deadline.
You are asked to set up a system for displaying the status of your programs in the form of debug messages controlled by the environment variable DEBUG_REST. If it exists, it must contain an integer. If it does not exist or if its value is 0, your programs should display a line at the maximum to indicate the result (except in case of error where you are not limited) as in the example in Annex. If the value equals 1, programs should display a short message during important steps.
With higher values, your programs should show more complete debugging information, including details of synchronizations. Do not forget to use **fflush** to have these messages as soon as they are display when output is redirected to a file.
A set of test sets is at your disposal. These constitute additional specifications, especially in terms of messages expected on the display. You should not modify these tests in any way, but you can possibly supplement them with new scripts. The coverage-and-testing target of Makefile can give you ideas for adding tests.
The scripts provided incorporate durations and are therefore sensitive to the speed of the computer running them.