TP

1 Introduction

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: reducing the size and keeping the contact details of customers (the guests) to identify contact cases. From the perspective of future deconfinement, the aim of this lab is to help restaurateurs to respect health rules.

To do this, we consider that a restaurant has a fixed number of tables, each with a maximum number of seats and can accommodate a group. For example, a restaurant can have 3 tables of 2, a table of 4 and a table of 6 places. The guests arrive one after the other, announcing themselves (for tracing contact cases); the first guest in a group must announce the number of people expected in the group, the following simply 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 restaurant operator 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 multiple times: when a group has finished their meal, the table becomes available again and the restaurateur can assign it to a new group.

2 Work To be Done :

We ask to realize the following programs in C language:

1. The program `restaurant` is active for the duration of the service. The first argument is the length of a meal (expressed in milliseconds to speed up testing and debugging sessions) and the following are table capacities. At the end of the service, the restaurateur displays the total number of guests and groups welcomed.
2. The program `convive` 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 them and the guest ends.
   - the following guests must announce as the second argument the name of the first guest in the group. If this is not present, the guest ends immediately as before.

   The guest remains active for the duration of the meal, which can only begin when the last member of the group has arrived or curfew time approaches. To simplify implementation, the end of meals is indicated by the restaurateur. When this signals the end of the meal, the guests can end.
3. The program `police`
4. simulates a control of the authorities. It should display the names of the guests currently 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.
5. The program `fermeture` indicates that the curfew hour is approaching: the restaurant owner must then refuse access to any new guest and start meals at incomplete tables. The closure program 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 the closure, they must be turned away.

When the last table has finished its meal, the program `restaurant` 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 several processes, you will only use shared memory POSIX, to the exclusion of any other mechanism such as file, pipe, etc. Likewise, for synchronization, you will only use semaphores POSIX, to the exclusion of any other mechanism such as barrier, signal, lock, etc.

3 Implémentation

The return code for your programs should indicate whether an error has occurred or not. In the event of an error, you will adopt the simple strategy of terminating the execution of the affected program with an explicit message. Refusal of a guest (no table, announced closing, etc.) should not be considered an error.

To simplify the implementation, we will assume that the following conditions are true. Unless stated otherwise, you are not asked to check 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 other than a number (to distinguish the second argument from `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 function `sem_timedwait` and read the manual carefully 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 debugging 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 must display at most one line to indicate the result (except in the event of an error where you are not limited) as in the example in the appendix.

If the value equals 1, programs should display a short message during important steps. With higher values, your programs should display more complete debugging information, including details of synchronizations. Remember to use `fflush` to have these messages as soon as they are displayed when the output is redirected to a file.

A set of test sets is at your disposal. These constitute additional specifications, in particular with regard to the messages expected on the display. You should not modify these tests in any way, but you can optionally supplement them with new scripts. Target `couverture-et-tests` of `Makefile` can give you ideas to add tests.

The scripts provided incorporate durations and are therefore sensitive to the speed of the computer running them. You can use the variable `MARGE` to act on certain tolerances, (for example: `MARGE=60 make test`), but success is by no means guaranteed.

Appendix: example session

The example below shows a fictitious session. Note that the programs are almost all started in the background (with &), but could advantageously be launched in different windows.

The test sets provided give other examples that you may find useful.

```
>  ./restaurant 3600000 2 4 2 4 2 &      # a meal lasts 1 hour, there are 5 tables (numbered 0 to 4)

>  ./convive Alceste 3 &                 # Alceste arrives and announces a group of 3 people
Welcome Alceste, you have the table 1 # the table 1 to 4 places (display by convive)
>  ./convive Celimene Alceste &          # Celimene arrives and joins Alceste
Welcome Celimene, you have the table 1

>  ./convive Sganarelle 2 &              # Sganarelle announces itself as a new group
Welcome Sganarelle, you have the table 0

>  ./convive Oronte Alceste &            # Oronte arrives and joins Alceste and Celimene
Welcome Oronte, you have the table 1     # The 3 have arrived, the meal can begin

>  ./convive Martine Sganarelle &        # Martine arrives and joins Sganarelle
Welcome Martine, you have the table 0 # The 2 have arrived, the meal can begin

>  ./police
table 0 : Sganarelle Martine            # police first check the current occupation
table 1 : Alceste Celimene Oronte
table 2 : (Empty)
...
reminder book:                          # ... then the reminder book provided by the restaurant owner
Groupe 1 : Alceste Celimene Oronte
Groupe 2 : Sganarelle Martine

# one hour after the arrival of Oronte, the meal of the 3 stops and the 3 "convive" programs end.
# Table 1 can be reused for another group
>  ./police                             # Again?
Table 0 : Sganarelle Martine
Table 1 : (Empty)
Table 2 : (Empty)
...
reminder book:
Groupe 1 : Alceste Celimene Oronte
Groupe 2 : Sganarelle Martine

>  ./convive Argand 5 &                 # No table available for 5 guests
Sorry Argand, no table available
>  ./convive Toinette Argand &          # Too bad Argand has already been turned away
Sorry Toinette, no Argand here

>  ./fermeture                          # the curfew is approaching: we must end the service

>  ./convive Jourdain 2                 # Too late!
Sorry Jourdain, no table available

# the restaurateur (program restaurant launched here in the background) waits for the guests currently
# at the table to finish their meal, then displays the service report and ends.
4 Guests served in 2 groupes
```