

TP 5 (à rendre)

Note : ce TP est un travail individuel, à rendre avant le 6 décembre 2020 à 20h (voir modalités plus bas).

1 Introduction

La situation économique est difficile en cette période d'incertitudes sanitaires. Les restaurateurs sont notamment soumis à rude épreuve en devant respecter des consignes strictes : réduction de la jauge et conservation des coordonnées des clients (les convives) pour identifier les cas contacts. Dans la perspective du déconfinement futur, le but de ce TP est d'aider les restaurateurs à respecter les règles sanitaires.

Pour ce faire, on considère qu'un restaurant dispose d'un nombre fixe de tables, chacune ayant un nombre maximum de places et pouvant accueillir un groupe. Par exemple, un restaurant peut avoir 3 tables de 2, une table de 4 et une table de 6 places. Les convives arrivent les uns après les autres en s'annonçant (pour le traçage des cas contacts) ; le premier convive d'un groupe doit obligatoirement annoncer le nombre de personnes prévu dans le groupe, les suivants se contentent d'indiquer la première personne du groupe. Lorsqu'un nouveau groupe est annoncé, le restaurateur alloue la plus petite table disponible pouvant accueillir le groupe.

À tout moment, il peut y avoir un contrôle des autorités. Comme le montant de l'amende est élevé, le restaurateur doit montrer qu'il respecte les règles : les autorités vérifient l'affectation des tables ainsi que la tenue du « cahier de rappels » où sont notées les convives et leur groupe.

Une table ne peut accueillir qu'un seul groupe à la fois, mais peut bien sûr servir plusieurs fois : lorsqu'un groupe a terminé son repas, la table redevient disponible et le restaurateur peut l'affecter à un nouveau groupe.

2 Travail à réaliser

On demande de réaliser les programmes suivants en langage C :

1. Le programme `restaurant` est actif pendant toute la durée du service. Le premier argument est la durée d'un repas (exprimé en millisecondes pour accélérer les sessions de test et de déboguage) et les suivants sont les capacités des tables. À la fin du service, le restaurateur affiche le nombre total de convives et de groupes accueillis.
2. Le programme `convive` simule un convive. Le premier argument est le nom du convive. Le deuxième argument a une signification différente pour le premier convive d'un groupe ou pour les suivants :
 - si le convive est le premier d'un groupe, il doit annoncer en deuxième argument le nombre de personnes attendues pour le groupe (y compris lui-même). S'il n'y a pas de table suffisante pour le groupe, le restaurateur l'en informe immédiatement et le convive se termine.
 - les convives suivants doivent annoncer en deuxième argument le nom du premier convive du groupe. Si celui-ci n'est pas présent, le convive se termine immédiatement comme précédemment.Le convive reste actif pendant toute la durée du repas, qui ne peut commencer que lorsque le dernier membre du groupe est arrivé ou que l'heure du couvre-feu approche. Pour simplifier l'implémentation, la fin des repas est indiquée par le restaurateur. Lorsque celui-ci signale la fin du repas, les convives peuvent se terminer.
3. Le programme `police` simule un contrôle des autorités. Il doit afficher les noms des convives actuellement présents à chaque table ainsi que l'état actuel du cahier de rappels, qui répertorie l'ensemble des groupes ayant été servis ainsi que ceux actuellement attablés.
4. Le programme `fermeture` indique que l'heure du couvre-feu approche : le restaurateur doit alors refuser l'accès de tout nouveau convive et commencer les repas des tables non complètes. Le programme `fermeture` ne fait qu'indiquer une situation au restaurateur et n'attend pas la fermeture effective : les groupes déjà installés (même si tous les membres ne sont pas encore arrivés) peuvent quand même terminer leur repas. Si un nouveau membre d'un groupe installé arrive après la fermeture, il doit être refoulé.

Lorsque la dernière table a terminé son repas, le programme `restaurant` peut alors se terminer et afficher le nombre de convives ainsi que le nombre de groupes servis.

Bien évidemment, vous éviterez toute attente active, même ralentie. Pour partager des informations entre plusieurs processus, vous n'utiliserez que la mémoire partagée POSIX, à l'exclusion de tout autre mécanisme tel que fichier, tube, etc. De même, pour la synchronisation, vous n'utiliserez que les sémaphores POSIX, à l'exclusion de tout autre mécanisme tel que barrière, signal, verrou, etc.

Vous rédigerez un rapport comprenant notamment la description des synchronisations (événement caractéristique de la synchronisation, acteurs concernés, mécanisme utilisé) ainsi que la justification des informations placées en mémoire partagée.

3 Implémentation

Le code de retour de vos programmes devra indiquer si une erreur s'est produite ou non. En cas d'erreur, vous adopterez la stratégie simple de terminer l'exécution du programme concerné avec un message explicite. Le refoulement d'un convive (pas de table, fermeture annoncée, etc.) ne doit pas être considéré comme une erreur.

Pour simplifier l'implémentation, on supposera que les conditions suivantes sont vérifiées. Sauf mention contraire, on ne vous demande pas de les vérifier vous-mêmes.

- Il n'y a qu'un seul restaurant présent dans le système.
- Les noms des convives sont limités à 10 caractères (limite à vérifier).
- Les noms des convives peuvent contenir n'importe quel caractère, mais doivent commencer par un caractère différent d'un chiffre (pour distinguer le deuxième argument de `convive`)
- Le nom d'un convive suffit pour l'identifier, autrement dit le nom est unique.
- La capacité maximum d'une table est de 6 convives (limite à vérifier).
- Le nombre de tables dans la salle et le nombre de groupes accueillis durant l'ouverture du restaurant ne sont pas limités.

Pour gérer la durée des repas, il est suggéré d'utiliser la fonction `sem_timedwait` et de lire attentivement le manuel pour la signification du paramètre indiquant la date limite.

On vous demande de mettre en place un système d'affichage de l'état de vos programmes sous forme de messages de débogage contrôlés par la variable d'environnement `DEBUG_REST`. Si celle-ci existe, elle doit contenir un entier. Si elle n'existe pas ou si sa valeur est 0, vos programmes doivent afficher une ligne au maximum pour indiquer le résultat (sauf en cas d'erreur où vous n'êtes pas limités) comme dans l'exemple en annexe. Si la valeur égale 1, les programmes doivent afficher un court message lors des étapes importantes. Avec des valeurs supérieures, vos programmes doivent afficher des informations de débogage plus complètes, notamment le détail des synchronisations. N'oubliez pas d'utiliser `fflush` pour avoir ces messages dès leur affichage lorsque la sortie est redirigée dans un fichier.

Un ensemble de jeux de tests est à votre disposition. Ceux-ci constituent des spécifications complémentaires, notamment en matière de messages attendus à l'affichage. Vous ne devez en aucun cas modifier ces tests, mais vous pouvez éventuellement les compléter avec de nouveaux scripts. La cible `couverture-et-tests` du `Makefile` peut vous donner des idées pour ajouter des tests.

Les scripts fournis incorporent des durées et sont donc sensibles à la vitesse de l'ordinateur qui les exécute. Tels quels, ils fonctionnent sur `turing.unistra.fr` qui sert de référence. Vous pouvez utiliser la variable `MARGE` pour agir sur certaines tolérances, (par exemple : `MARGE=60 make test`), mais la réussite n'est en aucun cas garantie.

4 Modalités de remise

Vous déposerez sur Moodle votre TP, avec votre rapport, sous forme d'une archive au format « tar.gz » dans l'espace de devoirs prévu à cet effet. Vous supprimerez les fichiers binaires et autres fichiers de log. Les rendus présentant de trop fortes similitudes seront sanctionnés.

Annexe : exemple de session

L'exemple ci-dessous illustre une session fictive. On notera que les programmes sont presque tous lancés en arrière plan (avec &), mais pourraient avantageusement être lancés dans des fenêtres différentes.

Les jeux de tests fournis donnent d'autres exemples qui peuvent vous être utiles.

```
> ./restaurant 3600000 2 4 2 4 2 &      # un repas dure 1 h, il y a 5 tables (numérotées 0 à 4)

> ./convive Alceste 3 &                  # Alceste arrive et annonce un groupe de 3 personnes
Bienvenue Alceste, vous avez la table 1 # la table 1 a 4 places (affichage par convive)
> ./convive Celimene Alceste &          # Celimene arrive et rejoint Alceste
Bienvenue Celimene, vous avez la table 1

> ./convive Sganarelle 2 &              # Sganarelle s'annonce comme un nouveau groupe
Bienvenue Sganarelle, vous avez la table 0

> ./convive Oronte Alceste &           # Oronte arrive et rejoint Alceste et Celimene
Bienvenue Oronte, vous avez la table 1   # Les 3 sont arrivés, le repas peut commencer

> ./convive Martine Sganarelle &       # Martine arrive et rejoint Sganarelle
Bienvenue Martine, vous avez la table 0  # Les 2 sont arrivés, le repas peut commencer

> ./police                                # Trffff ! Papiers siouplé !
table 0 : Sganarelle Martine             # police vérifie d'abord l'occupation actuelle
table 1 : Alceste Celimene Oronte
table 2 : (vide)

...
cahier de rappels :                      # ... puis le cahier de rappels fourni par le restaurateur
Groupe 1 : Alceste Celimene Oronte
Groupe 2 : Sganarelle Martine

# une heure après l'arrivée de Oronte, le repas des 3 s'arrête et les 3 programmes "convive" se terminent.
# La table 1 peut être réutilisée pour un autre groupe

> ./police                                # Encore ?
Table 0 : Sganarelle Martine
Table 1 : (vide)
Table 2 : (vide)

...
cahier de rappels :
Groupe 1 : Alceste Celimene Oronte
Groupe 2 : Sganarelle Martine

> ./convive Argand 5 &                   # Pas de table disponible pour 5 convives
Desole Argand, pas de table disponible
> ./convive Toinette Argand &           # Dommage, Argand a déjà été refoulé
Desole Toinette, pas de Argand ici

> ./fermeture                               # le couvre-feu approche : il faut terminer le service

> ./convive Jourdain 2                     # Trop tard !
Desole Jourdain, pas de table disponible

# le restaurateur (programme restaurant lancé ici en arrière plan) attend que les convives actuellement
# à table achèvent leur repas, puis affiche le bilan du service et se termine.
5 convives servis dans 2 groupes
```