NOM Prénom :

# CSC3102 – Contrôle Final 1
# Année 2022 – 2023

Consignes :

- Répondez aux questions en écrivant soit dans les cadres fournis au fil du sujet, soit dans les listings des scripts placés en fin d'exercice.
- Si vous manquez de place, vous pouvez écrire en fin de sujet **en l'indiquant dans l'encart initial**.
- Les réponses qui ne requièrent pas de code doivent être justifiées.
- Seules l'annexe et vos notes personnelles manuscrites sont autorisées.
- La durée est fixée à 1h30.
- Le barème est donné à titre indicatif.

## Petit tour de carrousel !

Il s'agit ici de simuler des usagers faisant des tours de manège. Pour cela, vous sont donnés les squelettes de scripts placés à la fin du sujet et à compléter au fil des questions :

- manege.sh simule un manège qui tourne dès qu'il est complet et ne ferme pas tant que des usagers ont des tickets ;
- usager.sh simule un usager doté de 4 tickets et qui les utilise sur un manège désigné en paramètre ;
- tour.sh simule un usager dans la file d'attente d'un manège qui, lorsqu'une place est disponible, prend place dans ce manège avant de donner un de ses tickets ;
- run.sh lance un scénario qui, d'une part, crée un caroussel à 3 places, et qui, d'autre part, crée 4 usagers utilisant ce caroussel.

Conseil : Dans cet exercice fil rouge, les questions ne se suivent pas forcément. N'hésitez pas à passer.

**[2pts] Préambule**

**1)** [0,5 pt] Donnez **la** commande qui permet de créer un répertoire nommé "Manege" dans votre répertoire de connexion quel que soit votre positionnement dans le système de fichiers.

```
mkdir ~/Manege
```

**2)** [0,5 pt] Donnez **la** commande qui déplace tous les scripts Shell du répertoire de travail, vers le répertoire "Manege" de votre répertoire de connexion.

```
mv *.sh ~/Manege
```

**3)** [1 pt] Donnez **la séquence** de commandes qui (1) change le répertoire de travail pour être "Manege" ; et (2) ajoute des droits d'exécution au propriétaire des scripts Shell (fichiers dont le suffixe est .sh) placés dans ce nouveau répertoire de travail **en utilisant un chemin relatif**.

```
cd ~/Manege
chmod u+x *.sh
```

**[3pts] Partie manège**

En lisant le script manege.sh, vous comprenez qu'il prend en entrée deux arguments : le nom du manège à simuler et le nombre de ses places.

Le manège démarre une fois que son nombre de places est enregistré dans un fichier de configuration nommé {nom du manege}.manege, où « {nom du manege} » est remplacé par son nom. Ce fichier permettra de suivre l'état d'occupation du manège.

(a) [1pt] Dans le script manege.sh, écrivez la portion de code qui fixe le nombre de places initial du manège dans son fichier de configuration.

Le manège entre ensuite dans une boucle infinie en trois phases : une phase d'attente d'installation de ses usagers, la mise en marche du manège et sa réinitialisation pour le tour suivant.

(b) [2pts] Dans le script manege.sh, écrivez la portion de code qui permet d'attendre que le manège soit complet. Pour cela, consultez le nombre de places disponibles depuis le fichier de configuration du manège ; puis, tant que ce nombre n'est pas égal à 0, mettez le manège en attente.

**[1pt] Partie usager**

En lisant le script usager.sh, vous comprenez qu'il prend en entrée deux arguments : le nom de l'usager et le nom du manège pour lequel il se dote de 4 tickets. Le nombre de tickets détenus par un usager est stocké dans un fichier de configuration nommé {nom de l'usager}-{nom du manege}.tickets, où « {nom de l'usager » et « {nom du manege} » sont remplacés par les noms respectifs.

L'usager commence par attendre l'ouverture du manège. Pour cela, il vérifie l'existence du fichier de configuration {nom du manege}.manege, associé au manège, qui a été créé dans le script manege.sh.

(c) [1pt] Dans le script usager.sh, écrivez la boucle qui attend l'ouverture du manège.

Il fait ensuite autant de tours que son quota de tickets le lui permet.

**[9 pts] Partie tour**

En lisant le script tour.sh, vous comprenez qu'il prend en entrée deux arguments : le nom d'un usager et le nom du manège sur lequel celui-ci veut faire un tour.
Si le manège est complet (c'est-à-dire si son nombre de places disponibles est égal à 0), l'usager est mis en attente. Sinon il s'installe dans le manège, en décrémentant le compteur de places libres du manège et en donnant un de ses tickets.

Le script run.sh orchestre le fonctionnement d'un manège et de ses usagers.

(d) [2pts] En l'état actuel des scripts manege.sh, usager.sh et tour.sh, l'exécution du script run.sh expose-t-elle des problèmes de concurrence ? Argumentez votre réponse dans tous les cas. En particulier si la réponse est oui, donnez un scénario d'exécution menant à un état incohérent. Attention à bien considérer le cas de chaque donnée partagée.

| |
|---|
| Pas de concurrence |

(e) [2pts] Si nécessaire, faites des corrections pour que le programme soit correct du point de vue de la concurrence. Vous pouvez supposer que vous avez à votre disposition les scripts P.sh et V.sh vus en cours. À cette question, run.sh ne doit pas être modifié.

(f) [1pt] Dans le script run.sh, modifiez le lancement des usagers de manière à les lancer en arrière plan.

(g) [2pts] À présent, l'exécution du script run.sh engendre-t-elle des problèmes de concurrence ? Argumentez votre réponse dans les deux cas. En particulier, si la réponse est oui, donnez un scénario d'exécution menant à un état incohérent. Attention à bien considérer le cas de chaque donnée partagée.

Concurrence sur le nombre de places libres dans le manège.

(h) [2pts] Si nécessaire, faites des corrections pour que le programme soit correct du point de vue de la concurrence. Vous pouvez supposer que vous avez à votre disposition les scripts P.sh et V.sh vus en cours. À cette question, run.sh ne doit pas être modifié.

**[5 pts] Partie orchestration**

Vous avez déjà pris en main le script run.sh, qui a pour rôle de lancer le manège et ses usagers. Cependant, il ne gère pas leur terminaison, ce qui est problématique pour le manège qui n'a aucun moyen de sortir de sa boucle infinie. Ainsi, dans cette partie, il s'agit de mettre fin au processus manege.sh une fois que tous les processus usager.sh lancés depuis run.sh sont terminés.

(i) [1pt] Dans le script usager.sh (après l'ouverture du manège mais avant de monter dans le manège), enregistrez ce processus usager.sh comme étant usager d'un manège. Pour ce faire, ajoutez son PID à la fin d'un nouveau fichier, nommé {nom du manege}.usagers.

(j) [2 pts] À la fin du script run.sh, attendez la terminaison de l'ensemble des processus dont le PID est enregistré dans le fichier {nom du manege}.usagers.

(k) [1 pt] À la suite dans le script run.sh, envoyez un signal USR1 au processus manege.sh dont le PID est stocké dans la variable pid_manege.

(l) [1 pt] Dans le script manege.sh, mettez en place un gestionnaire de signal adapté à la réception du signal envoyé à la question précédente. Le gestionnaire de signal doit afficher le message « {nom du manege} ferme ! », avec « {nom du manège} » remplacé par le nom du manège, et arrêter le processus.

**manege.sh**

```
#! /bin/sh

if [ $# -ne 2 ]; then
  echo "README - manege M P"
  echo "Un manège M tourne si ses"
  echo "P places sont prises."
  echo "M = nom (un mot) du manège"
  echo "P = places par tour du manège"
  exit 1
fi

#(l) fermeture du manège

trap 'echo "$1 ferme !"; exit 0' USR1



#(a) initialisation du nombre de places

echo "$2" > "$1.manege"



echo "Manège $1 ouvert!"



while true; do


  #(b) installation des usagers

  read libres < "$1.manege"
  while [ "$libres" -gt 0 ]; do
      sleep 1
      read libres < "$1.manege"
  done


  # Le manège tourne.
  echo "Tourne, tourne joli manège !"
  sleep 2


  # Réinitialisation du manège.


  echo $2 > $1.manege


done
```

**usager.sh**

```
#! /bin/sh

if [ $# -ne 2 ]; then
  echo "README - usager.sh U M"
  echo "Un usager U profite du"
  echo "manège M jusqu'à épuisement"
  echo "de ses tickets."
  echo "U = nom (un mot) de l'usager"
  echo "M = nom (un mot) du manège"
  exit 1
fi

nb_tickets=4


echo $nb_tickets > $1-$2.tickets


#(c) attente de l'ouverture du manège

while [ ! -f "$2.manege" ]; do
    sleep 1
done




#(i) enregistrement sur le manège

echo $$ >> "$2.usagers"



# Montée dans le manège :
# enchaînement des tours.
while [ $nb_tickets -ne 0 ]; do


  ./tour.sh $1 $2


  echo "$1 a fait un tour sur $2."


  read nb_tickets < $1-$2.tickets


done


echo "$1 quitte le manège $2."
```

**tour.sh**

```
#! /bin/sh

if [ $# -ne 2 ]; then
  echo "README - tour.sh U M"
  echo "Un usager U attend pour le"
  echo "manège M. Lorsque vient son"
  echo "tour, il donne un ticket pour"
  echo "s'installer dans le manège."
  echo "U = nom (un mot) de l'usager"
  echo "M = nom (un mot) du manège"
  exit 1
fi

# File d'attente.

#(e) pas de concurrence à corriger
#(h) correction de la concurrence

./P.sh "$2.lock"
read places < $2.manege



while [ $places -le 0 ]; do

  ./V.sh "$2.lock"

  sleep 1

  ./P.sh "$2.lock"

  read places < $2.manege



done

expr $places - 1 > $2.manege

./V.sh "$2.lock"

# Remise du ticket.



read tickets < $1-$2.tickets



expr $tickets - 1 > $1-$2.tickets
```

**run.sh**

```
#! /bin/sh




manege=Carrousel




./manege.sh $manege 3 &
pid_manege=$!



for i in alex gloria marty melman; do

  #(f) lancement en arrière-plan

  ./usager.sh $i $manege &

done



# Attendre le fichier $manege.usagers
sleep 10



#(j) attente de la terminaison des
usagers

usagers="$(cat "$manege.usagers")"
wait $usagers



#(k) signaler la terminaison au
manège

kill -USR1 $pid_manege
```

*page blanche à usage libre*