NOM Prénom :

CSC3102 – Contrôle Final 2
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 chenille !

Cet exercice simule l'enchaînement de plusieurs tours d'un manège nommé Chenille.
À cette fin, trois scripts sont mis à disposition en fin de sujet à compléter au fil des questions et de leur balisage dans le code :

- manege.sh simule un manège qui s'élance dès que l'ensemble de ses places sont occupées ;
- usager.sh simule un usager qui prend place dans un manège ;
- chenille.sh crée un manège nommé Chenille, puis lance autant d'usagers que nécessaire afin de remplir la chenille pour un nombre de tours tiré aléatoirement.

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

[7,5pts] Préambule

Pour l'instant, préparons l'environnement de travail.

1)[0,5pt] Sans vous déplacer dans l'arborescence, donnez **la** commande qui permet de créer un répertoire nommé "Manege" dans le répertoire CSC3102 pré-existant dans votre répertoire de connexion grâce à un chemin absolu.

2)[0,5pt] Donnez **la** commande qui permet de consulter votre positionnement dans le système de fichiers.

3)[1pt] Sans changer de position, donnez **la** commande qui déplace tous les scripts Shell (fichiers font le suffixe est .sh), du répertoire CF2 de votre répertoire de travail, dans le répertoire créé à la question 1.

4)[0,5pt] Donnez la commande qui permet de vous déplacer dans le répertoire créé à la question 1.

5)[1pt] Donnez **la** commande qui permet de lister les entrées du répertoire de travail commençant par CSC.

6)[0,5pt] Donnez la commande qui permet de compter le nombre d'entrées contenues dans le répertoire de travail et qui commencent par CSC.

7)[0,5pt] Donnez la commande qui permet d'écrire le nom des entrées du répertoire de travail commençant par CSC dans un fichier nommé csc.txt.

8)[1pt] Sans vous déplacer, donnez **la** commande qui duplique le répertoire Manege en Manege.sauv dans le même dossier. Attention à votre positionnement actuel par rapport au dossier Manege.

9)[1,5pt] Donnez la séquence de commandes qui permet (1) de vous déplacer dans le répertoire Manege.sauv créé à la question précédente ; (2) de renommer les entrées listées dans le fichier csc.txt de la question 7 sous le même nom, mais suffixé par « .sauv ».

10)[0,5pt] Donnez **la** commande qui retire les droits d'exécution à tout utilisateur des scripts Shell (fichiers dont le suffixe est .sh) placés dans le répertoire de travail.

[4pts] Manège générique

Le script manege.sh simule **un manège dont le nom est donné en premier et unique paramètre**, et dont le nombre de places disponibles est fixé par tirage aléatoire. Familiarisez-vous avec la structure du script. Dans cette partie, il vous sera demandé de diffuser ce nombre de deux manières différentes :
(a) au travers d'un **tube nommé**, afin d'en informer le processus qui a demandé la création du manège (le script chenille.sh) pour qu'il lance un nombre d'usagers adéquats ;
(b) grâce à un **fichier partagé**, qui indiquera à tout moment le nombre de places qui restent disponibles au fur et mesure du remplissage du manège par les usagers lors d'un tour.

(a) [2pts] Le nombre de places disponibles dans le manège est diffusé au travers d'un tube nommé. Ainsi, dans le script manege.sh à la balise (a), écrivez la portion de code qui (1) crée un canal de communication nommé {nom du manege}.pipe, où « {nom du manege} » est remplacé par son nom ; (2) configure l'utilisation de ce canal en lecture/écriture ; (3) écrit dans ce canal le nombre de places disponibles, qui est stocké dans la variable nb_places.

(b) [0,5pt] Afin de maintenir la cohérence du nombre de places disponibles entre les usagers du manège, celui-ci est enregistré dans un fichier de configuration nommé {nom du manege}.manege, où « {nom du manege} » est remplacé par son nom. Ainsi à l'ouverture du manège, le nombre de places est initialisé dans ce fichier au nombre de places total du manège ; et sera réinitialisé de la même façon à chaque fin de tour de manège. Dans le script manege.sh aux balises (b1) et (b2), écrivez la même portion de code qui fixe le nombre de places initial du manège dans son fichier de configuration.

(c) [1,5pts] Une fois le manège initialisé et ouvert, il entre dans une boucle infinie en trois phases : l'attente d'installation de ses usagers, sa mise en marche, et sa réinitialisation pour le tour suivant. Dans le script manege.sh à la balise (c), écrivez la portion de code qui permet d'attendre que le manège soit complet. Pour cela, (1) consultez le nombre de places disponibles depuis le fichier de configuration du manège ; tant que ce nombre est supérieur à 0, (2) mettez le manège en attente pendant 1 seconde, et (3) consultez à nouveau le nombre de places disponibles.

[8,5pts] Chenille

Le script chenille.sh permet la simulation d'un manège, dont le nom « chenille » est stocké dans la variable manege, et d'un ensemble d'usagers y prenant place. Familiarisez-vous avec la structure du script.

(d) [1,5pts] Dans le script chenille.sh à la balise (d), (1) lancez en arrière-plan un processus manege.sh simulant un manège nommé par le contenu de la variable manege, et (2) sauvegardez l'identifiant de ce processus dans une variable pid_manege.

(e) [0,5pt] Une fois lancé, en balise (e), écrivez la commande qui permet de lire dans la variable nb_places, sur le tube nommé ouvert par le manège, son nombre de places disponibles par tour.

(f) [2pts] À la balise (f), tous les usagers du jour sont lancés simultanément (script usager.sh).
Vous remarquez que les identifiants des processus usager.sh sont stockés dans un fichier nommé {nom du manege}.usagers, où « {nom du manege} » est remplacé par son nom stocké dans la variable manege ; cette remarque sera utile pour la question h.
En l'état, ces lancements successifs exposent-ils des problèmes de concurrence ? Argumentez votre réponse dans tous les cas. Si la réponse est oui, donnez un scénario d'exécution menant à un état incohérent.

(g) [2pts] Si nécessaire, faites des corrections dans le(s) script(s) manege.sh ou usager.sh, 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.

(h) [1pt] La balise (h) atteinte, il s'agit maintenant d'attendre la terminaison des processus usager.sh afin de simuler le fait que tous aient profité de leur tour de manège. Écrivez la portion de code qui permet d'attendre l'ensemble des processus dont l'identifiant a été stocké dans le fichier {nom du manege}.usagers.

(i) [1,5pts] Il reste à notifier son arrêt au manège. Pour cela, (1) en balise (i) du script chenille.sh, lancez un signal USR1 au processus simulant la fermeture du manège ; et (2) mettez en place le traitement de ce signal en balise (i) dans le script manege.sh : à sa réception, affichez un message indiquant la fermeture du manège, supprimez les fichiers relatifs au manège (créés aux questions a et b), et mettez un terme au processus courant.

manege.sh

```
#! /bin/sh

if [ $# -ne 1 ]; then
  echo "README - manege M avec M = nom (un mot) du manège"
  exit 1
fi

#(i) réception du signal de terminaison




#(a) initialisation du nombre de places du manege
nb_places=$(expr $(expr $RANDOM % 10) + 1)




#(b1) initialisation du manege




echo "Manège $1 ouvert!"

while true; do
  #(c) installation des usagers




  # Le manège tourne.
  echo "Tourne, tourne joli manège !"
  sleep 5

  #(b2) réinitialisation du manège.



done
```

chenille.sh

```
#! /bin/sh

manege=chenille

#(d) lancement du manege "chenille" ...



#(d) ... et sauvegarde de son pid



# attente de la création du manège
while ! [ -f $manege.manege ]; do
    sleep 1
done

#(e) recup nb_places




# tirage aléatoire du nombre de tours
nb_tours=$(expr $(expr $RANDOM % 5) + 1)

echo "$manege tourne $nb_tours fois avec $nb_places places disponibles"

nb_usagers=$(expr $nb_places \* $nb_tours)

#(f) création des usagers
while [ ! $nb_usagers -eq 0 ] ; do
    ./usager.sh $manege &
    echo $! >> $manege.usagers
    nb_usagers=$(expr $nb_usagers - 1)
done

#(h) attente de terminaison des usagers









rm $manege.usagers

#(i) lancement du signal de terminaison du manège



echo "FIN"
```

usager.sh

```
#! /bin/sh

if [ $# -ne 1 ]; then
  echo "README - usager.sh M"
  echo "M = nom (un mot) du manège"
  exit 1
fi




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



read places < $1.manege



while [ $places -le 0 ]; do



  sleep 1



  read places < $1.manege



done



echo "Un usager prend place dans $1"



expr $places - 1 > $1.manege
```