NOM Prénom :

CSC3102 – Contrôle Final 2
Année 2023 – 2024

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, aux endroits des marques correspondant aux questions ;
- si vous manquez de place, vous pouvez écrire sur la page 5 **en l'indiquant dans l'encart initial ;**
- les réponses qui ne requièrent pas de code doivent être justifiées ;
- polycopié, notes personnelles manuscrites et exemples de script sont autorisés ;
- la durée est fixée à 1h30 ;
- le barème est donné à titre indicatif.

1)[0,5pt] Donnez **la** ligne de commande qui permet de créer un répertoire nommé *CF2* dans votre répertoire de connexion **quel que soit** votre positionnement dans le système de fichiers.

```
mkdir ~/CF2
```

2)[0,5pt] Donnez **la** ligne de commande qui vous place dans le répertoire *CF2* de votre répertoire de connexion **quel que soit** votre positionnement dans le système de fichiers.

```
cd ~/CF2
```

3)[2.5 pts] Donnez la suite de commandes qui permet d'attendre la terminaison d'un ensemble de processus dont les PIDs sont stockés, un par ligne, dans un fichier nommé pids_en_attente.data.

```
while read $pid ; do
   pids= « $pids $pid »
done < pids_en_attente.data

wait $pids
```

4)[0.5 pt] Donnez une commande qui permet de lister l'ensemble des processus actifs.

```
pstree ou top ou ps
```

5)[1 pt] Donnez la commande qui permet de dupliquer le contenu de votre répertoire de travail dans le répertoire Sauvegarde du répertoire parent de votre répertoire de travail.

```
cp -r * ../Sauvegarde
```

Cérémonie d'ouverture des olympiades de l'INT

Afin d'obtenir un accès gratuit à la cérémonie d'ouverture des olympiades de l'INT, les spectateurs sont tenus de faire une demande auprès d'un guichet de réservation. Ce guichet unique collecte les demandes, qui sont par la suite traiter par un ensemble de serveurs de réservation.

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

[4 pts] Guichet de réservation

Le guichet de réservation a pour rôle de collecter les demandes de réservation jusqu'à épuisement des places disponibles selon un mode premier arrivé, premier servi. Une fois sa phase d'initialisation réalisée, le script *guichet.sh* entre dans une boucle infinie qui réceptionne les demandes de réservation, les acquitte s'il reste des places disponibles ou les décline sinon.

Dans le script *guichet.sh ,*

(a) [0,5pt] Déclarez une variable *nb_places* et initialisez-la à 150.

(b)[0,5pt] Mettez en place un canal de communication nommé guichet.*pipe* et maintenez-le ouvert en lecture et écriture. C'est sur ce canal que les spectateurs font leur demande de réservation.

Cette première partie d'initialisation faite (elle sera complétée dans la partie relative au serveur de réservation), passons aux différentes étapes réalisées par le guichet en vue de traiter une demande de réservation.

Dans une boucle infinie,

- **(c)**[0,5pt] le guichet attend les demandes de réservation en récupérant le PID d'un spectateur en demande de réservation sur le canal de communication du guichet,
- **(d)** [1pt]s'il reste des places,
  - inscrit le spectateur retenu en ajoutant son PID au fichier nommé *resa_INT.data,*
  - décrémente le nombre de places disponibles,
  - confirme au spectateur sa réservation en indiquant « OK » sur le canal de communication identifié du spectateur nommé *<pid>.pipe,* selon le PID du spectateur et créé dans le script *spectateur.sh*;
- **(e)**[0,5pt] sinon
  - indique que l'évènement est complet en indiquant « NOK » sur le canal de communication identifié du spectateur nommé *<pid>.pipe,* selon le PID du spectateur et créé dans le script *spectateur.sh*.

[1.5 pts] Spectateurs

Chaque spectateur fait une demande de réservation grâce à l'appel au script *spectateur.sh.* Son comportement se résume à l'ouverture d'un canal de communication nommé <pid.>.pipe selon le PID du processus courant, l'envoi de son PID sur le canal de communication *guichet.pipe* ouvert par *guichet.sh* et la récupération de la réponse sur son canal de communication propre. Si le spectateur est retenu, il se met en attente. Le code correspondant est donné en fin de sujet.

(f)[1.5pt] Existe-il des accès concurrents problématiques entre les spectateurs et le guichet ? Si oui, proposez un scénario d'exécution pathologique et protégez les sections critiques grâce aux scripts *P.sh* et V.sh utilisés en TP en les faisant précéder de la balise (f). Dans tous les cas, justifiez votre réponse.

Pas de concurrence car le guichet communique avec les spectateurs avec un pipe et pas de concurrence non plus (pour l'instant) sur le fichier resa_INT.data car le guichet est unique.

[pts] Serveur de réservation

Un serveur de réservation simule l'attribution d'une place précise aux spectateurs ayant été retenu par le guichet de réservation. Pour cela, il prend en charge le premier spectateur inscrit dans le fichier *resa_INT.data*, le retire de la liste et le notifie de la finalisation de sa résrevation.

Dans *serveur.sh*, dans une boucle infinie,

(g) [0,5pt] Faites une lecture de la première ligne du fichier *resa_INT.data* dans une variable nommée *spectateur.*

(h) [0,5pt] Si la chaîne de caractères lue est vide, attendez 3s avant de réitérer la lecture du fichier.

Sinon,

1. Retirez le client du fichier des clients en attente en suivant les étapes suivantes :

(i)[1,5pt] Commencez par déterminer le nombre de lignes du fichier *resa_INT.data* et stockez le résultat dans une variable *nb_lignes*. Pour cela, construisez incrémentalement la commande sur votre brouillon suivant les instructions suivantes avant de la reporter sur votre copie.

- Comptez le nombre de lignes du fichier resa_INT.*data* grâce à l'option -l de la commande *wc*.

- La commande produit un résultat dont le format est de la forme :

```
$ wc -l fic.data
2 fic.data
$
```

Sur une même ligne, il indique le nombre de lignes du fichier et rappelle le nom du fichier traité.
Ainsi, complétez votre commande en ne conservant que le premier champ du résultat produit par la commande précédente en prenant le soin d'utiliser le caractère « espace » comme délimiteur. La page de manuel de la commande *cut* vous permettrait de savoir que l'option *-f LISTE* (f pour *field)* permet de sélectionner le champ dont le numéro est listé et l'option -d, d'indiquer un nouveau délimiteur (celui par défaut est la tabulation).

- Stockez le résultat de la commande obtenue dans la variable *nb_lignes*.

(j)[1pt] Retirez le spectateur en tête du fichier des clients en attente. Vous pouvez utiliser des solutions déjà vue en TP, ou bien utiliser le résultat de l'étape (i) en suivant les étapes suivantes :

- Utilisez la commande *tail* (dont la page de manuel est disponible en fin de sujet) pour ne garder que les nb_lignes -1 dernières lignes du fichier resa_INT.*data ;*
- Écrivez le résultat de la commande dans un fichier resa_INT.*data.new* ;
- Renommez le fichier resa_INT.*data.new* en resa_INT.*data*.

2. **(k)** [1pt] Notifiez à l'aide du signal USR1 le processus spectateur pris en charge afin qu'il sorte de son état d'attente.

(l) [1pt] Dans le script approprié, mettez en place de traitement de cette notification afin de mettre fin à l'état d'attente du spectateur qui reçoit une notification telle que faite en question (k).

(m) [0.5 pt] Mettez à présent en place les serveurs qui attribueront concrètement les places aux spectateurs sélectionnés par le guichet. Dans le script *guichet.sh*, à la balise (m), avant de rentrer dans la boucle d'accueil, lancez deux serveurs en tâche de fond.

Mise en exécution

(n) [1pt] Positionnez correctement les droits nécessaires à l'exécution des scripts.

Terminal 1 :

```
chmod u+x *.sh
```

(o) [1 pt] Lancez un guichet et quatre spectateurs voulant faire une demande de réservation.

Terminal 2 :

```
./guichet.sh
```

Terminal 3 :

```
./spectateur.sh &
./spectateur.sh &
./spectateur.sh &
./spectateur.sh &
```

(o) [1,5 pts] Ce scénario d'exécution provoque-t-il des problèmes de concurrence. Si oui, donnez un scénario d'ordonnancement pathologique ; protégez les sections critiques identifiées dans les scripts à l'aide des scripts P.sh et V.sh, vus en TP et disponibles dans le répertoire où vous travaillez. Faites précéder vos ajouts de la balise **(o)**.

Concurrence sur resa_INT.data qui est modifié par guichet.sh et les serveurs.

guichet.sh

```
#! /bin/bash

#(a)
nb_places=150


#(b)
mkfifo guichet.pipe
exec 3<>guichet.pipe


#(m)
touch resa_INT.data
./serveur.sh &
./serveur.sh &


while true ; do

    #(c)
    read pid < guichet.pipe




    #(d)
    if [ $nb_places -gt 0 ]; then

     ./P.sh resa.lock
     echo $pid >> resa_INT.data
     ./V.sh resa.lock
     nb_places=$(expr $nb_places - 1)
     echo "OK" > $pid.pipe

    else
     #(e)
     echo "NOK" > $pid.pipe


    fi
done
```

spectateur.sh

```
#!/bin/bash

#(l)
trap 'exit 0' USR1




mkfifo $$.pipe
exec 3<>$$.pipe



echo $$ > guichet.pipe


read reponse < $$.pipe



if [ $reponse = "OK" ]; then

    while true ; do

     sleep 5

    done



fi
```

serveur.sh	

```
#!/bin/bash




while true ; do


    ./P.sh resa.lock
    #(g)

    read spectateur < resa_INT.data

    if [ ! -z $spectateur ]; then
     #(i)
     nb_lignes=$(wc -l resa_INT.data |
cut -d ' ' -f 1)


     #(j)
     tail -n $(expr $nb_lignes - 1)
resa_INT.data > resa_INT.data.new
     mv resa_INT.data.new resa_INT.data

     ./V.sh resa.lock



     echo "Attribution de places au
spectateur $spectateur"


     #(k)
     kill -USR1 $spectateur



    else
     #(h)
     sleep 3
     ./V.sh resa.lock









    fi
done
```

Annexe : page de manuel de la commande tail (extraits choisis)

NOM

tail - Afficher la dernière partie de fichiers

SYNOPSIS

tail [OPTION]... [FICHIER]...

Afficher les 10 dernières lignes de chaque FICHIER sur la sortie standard. Lorsqu'il y a plus d'un FICHIER, faire précéder chaque groupe de lignes d'un en-tête donnant le nom du fichier. L'entrée standard est lue quand FICHIER est omis ou quand FICHIER vaut « - ».

Les paramètres obligatoires pour les options de forme longue le sont aussi pour les options de forme courte.

-c, **--bytes**=K
afficher les K derniers octets ; vous pouvez aussi utiliser **-c** +K pour afficher les octets de chaque fichier à partir du Kième octet

-f, **--follow[=**{name|descriptor}]
afficher les dernières données ajoutées au fur et à mesure que le fichier s'accroît ; **-f**, **--follow**, et **--follow**=descriptor sont équivalents

-F identique à **--follow**=name **--retry**

-n, **--lines**=K
afficher les K dernières lignes, au lieu des 10 dernières ; ou utilisez **-n** +K pour afficher toutes les lignes à partir de la Kième.

--max-unchanged-stats=N
avec l'option **--follow**=name, réouvrir le FICHIER dont la taille n'a pas changé après N itérations (par défaut 5), afin de vérifier s'il a été détruit ou renommé (c'est habituellement le cas des fichiers journaux dont on effectue la rotation)

--pid=PID
avec **-f**, terminer après l'arrêt du processus PID

-q, **--quiet**, **--silent**
ne jamais afficher les en-têtes donnant les noms de fichiers

Si le premier caractère parmi K (nombre d'octets ou de lignes) est un « +», l'affichage commence à partir du Kième élément depuis le début de chaque fichier, sinon, les K derniers éléments du fichier sont affichés. K peut avoir un suffixe multiplicateur : b pour 512, kB pour 1000, K pour 1024, MB pour 1000*1000, M pour 1024*1024, GB pour 1000*1000*1000, G pour 1024*1024*1024, et ainsi de suite pour T, P, E, Z et Y.