Nom Prénom :

CSC3102 – Contrôle Final 1 (1h30)
Année 2021-2022

Consignes :

- Répondez aux questions en écrivant directement dans les cadres fournis au fil du sujet ou dans les listings des scripts placés en fin d'exercice. Si vous deviez manquer de place, vous pouvez également écrire en fin de sujet en l'indiquant dans l'encart initial.
- Sont autorisés le polycopié, l'annexe et les travaux pratiques du module, ainsi que vos notes personnelles manuscrites.
- La durée est fixée à 1h30.
- Le barème est donné à titre indicatif.

1) [1pt] Donnez LA commande qui permet de créer un répertoire nommé "Pschit" dans votre répertoire de connexion quel que soit votre positionnement dans le système de fichiers.

2) [1pt] Donnez LA commande qui permet d'afficher la liste des fichiers normaux dont le nom commence par "Ps" dans l'ensemble de l'arborescence de votre compte utilisateur.

3) [2pts] Donnez la chaîne de commandes qui permet de compter le nombre de lignes où apparaît "coucou" dans le fichier salutations.txt se trouvant dans le répertoire parent de votre répertoire de travail.

4) [1pt] Donnez les commandes qui permettent d'écrire "Hi all!" :

a) sur le terminal
b) dans le fichier salutations.txt du répertoire de travail.

5) [1pt] Donnez les commandes qui permettent de lancer deux processus xeyes en parallèle.

6) [1pt] Donnez la commande qui donne uniquement les droits d'écriture sur fichier.txt à tout utilisateur.

7) [3pts] Donnez la suite de commandes qui permet de scinder le fichier complet.txt en deux fichiers contenant le même nombre de lignes, nommés moitie1.txt et moitie2.txt. Pour cela, commencez par déterminer le nombre de lignes de chaque fichier en divisant par deux le nombre de lignes du fichier complet.txt; puis écrivez les x premières lignes dans le fichier moitie1.txt grâce à la commande head (page

de manuel à la fin du sujet), les x dernières lignes dans le fichier moitie2.txt grâce à la commande tail (page de manuel à la fin du sujet) et supprimez le fichier complet.txt.

8) [7pts] Une bibliothèque propose une liste de livres accessibles en ligne grâce à un fichier partagé nommé "bibliotheque.txt" (chemin vers un livre par ligne).

(a) [2pts] Écrire un script lecteur.sh qui choisit un livre dans la bibliothèque et le lit. Pour cela, tirez au sort un nombre correspondant à la ligne du livre choisi en utilisant la commande suivante : n=$(expr $RANDOM % $(wc -l bibliotheque.txt)) , lisez le fichier bibliotheque.txt jusqu'à la n-ième ligne afin de connaître le chemin vers le titre élu pour enfin afficher le contenu du fichier.

(b) [1pt] L'accès simultané à la bibliothèque par plusieurs lecteurs pose-t-il des problèmes de concurrence ? Si oui, donnez un scénario d'ordonnancement menant à un état incohérent et protégez les sections critiques grâce aux scripts P.sh et V.sh utilisés en cours.

(c) [1pt] La bibliothèque se met régulièrement à jour et ajoute des titres au fichier. Sans vous soucier de vérifier la présence de paramètre, écrivez le script bibliothecaire.sh qui ajoute à la fin du fichier bibliotheque.txt le chemin vers le livre donné en premier paramètre.

(d) [1pt] L'utilisation en parallèle d'un lecteur.sh et d'un bibliothecaire.sh pose-t-elle des problèmes de concurrence ? Si oui, donnez un scénario d'ordonnancement menant à un état incohérent et protégez les sections critiques grâce aux scripts P.sh et V.sh utilisés en cours.

(e) [2pts] L'utilisation en parallèle de plusieurs lecteur.sh et de plusieurs bibliothecaire.sh pose-t-elle des problèmes de concurrence ? Si oui, donnez un scénario d'ordonnancement menant à un état incohérent et protégez les sections critiques grâce aux scripts P.sh et V.sh utilisés en cours.

(f) [2pts] Un système automatisé vient supplanter le bibliothécaire : au lieu de lire le chemin vers le livre dans son premier paramètre, le script bibliothecaire.sh attend désormais de recevoir le chemin vers un nouveau livre sur un tube nommé d'après le PID du processus bibliothécaire ouvert en lecture grâce à une redirection avancée (commande exec). Modifiez le script en conséquence. L'opération d'ajout reste inchangée.

9) [4pts] Soit le script player.sh suivant qui émule la fonction de lecture d'un lecteur de CD qui joue en boucle le même morceau. play_morceau.sh bloque tant que le morceau en cours n'est pas terminé. On souhaite pouvoir passer au morceau suivant et arrêter la lecture en cours.

a) [1pt] Écrivez un script next.sh permettant de signaler au player dont le PID est en paramètre qu'il ne lira pas le morceau suivant une fois le morceau en cours achevé mais le suivant du suivant. Pour cela, vous devez incrémenter le compteur du morceau à venir.
b) [1pt] Complétez player.sh afin qu'il tienne compte de cette information si nécessaire.
c) [1pt] Écrivez un script stop.sh permettant de notifier au player qu'il doit s'arrêter.
d) [1pt] Complétez player.sh afin qu'il tienne compte de cette information si nécessaire.

player.sh

```
#! /bin/bash
echo "Let's play $$!"
morceau=1




while true; do



  ./play_morceau.sh $morceau



done
```

next.sh	stop.sh

bibliotheque.txt

/media/CarminaBurana.livre
/home/bob/Quejet’aime.livre
/home/poulet/AllezOnDanse.livre
/media/fiesta/Clafête.livre

lecteur.sh

! /*bin/bash*

bibliothecaire1.sh

! /*bin/bash*

bibliothecaire2.sh

! /*bin/*bash

HEAD(1) User Commands HEAD(1)

NAME head - output the first part of files

SYNOPSIS head [OPTION]... [FILE]...

DESCRIPTION

Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.

-c, --bytes=[-]NUM
print the first NUM bytes of each file; with the leading '-', print all but the last NUM bytes of each file

-n, --lines=[-]NUM
print the first NUM lines instead of the first 10; with the leading '-', print all but the last NUM lines of each file

-q, --quiet, --silent never print headers giving file names

-v, --verbose always print headers giving file names

-z, --zero-terminated line delimiter is NUL, not newline

--help display this help and exit

--version
output version information and exit

NUM may have a multiplier suffix: b 512, kB 1000, K 1024, MB 1000*1000, M 1024*1024, GB 1000*1000*1000, G 1024*1024*1024, and so on for T, P, E, Z, Y.

AUTHOR Written by David MacKenzie and Jim Meyering.

TAIL(1) User Commands TAIL(1)

NAME tail - output the last part of files

SYNOPSIS tail [OPTION]... [FILE]...

DESCRIPTION

Print the last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input. Mandatory arguments to long options are mandatory for short options too.

-c, --bytes=[+]NUM output the last NUM bytes; or use -c +NUM to output starting with byte NUM of each file

-f, --follow[={name|descriptor}] output appended data as the file grows;
an absent option argument means 'descriptor'

-F same as --follow=name –retry

-n, --lines=[+]NUM output the last NUM lines, instead of the last 10; or use -n +NUM to output starting with line NUM

--max-unchanged-stats=N with --follow=name, reopen a FILE which has not changed size after N (default 5) iterations to see if it has been unlinked or renamed (this is the usual case of rotated log files); with inotify, this option is rarely useful

--pid=PID with -f, terminate after process ID, PID dies

-q, --quiet, --silent never output headers giving file names

--retry keep trying to open a file if it is inaccessible

-s, --sleep-interval=N with -f, sleep for approximately N seconds (default 1.0) between iterations; with inotify and --pid=P, check process P at least once every N sec.

-v, --verbose always output headers giving file names

-z, --zero-terminated line delimiter is NUL, not newline

NUM may have a multiplier suffix: b 512, kB 1000, K 1024, MB 1000*1000, M 1024*1024, GB 1000*1000*1000, G 1024*1024*1024, and so on for T, P, E, Z, Y.

With --follow (-f), tail defaults to following the file descriptor, which means that even if a tail'ed file is renamed, tail will continue to track its end. This default behavior is not desirable when you really want to track the actual name of the file, not the file descriptor (e.g., log rotation). Use --follow=name in that case. That causes tail to track the named file in a way that accommodates renaming, removal and creation.