TELECOM SudParis

Contrôle de connaissances 2015/2016
des étudiants de 1ère année (EI1)

CSC3102 : Contrôle Final Session 2 - Sujet 1

Date : mardi 12 avril 2016 — Durée : 1 heure 30

Coordonnateurs : E. Brunet et G. Thomas

Documents autorisés : polycopié papier et notes manuscrites

Consignes :

- Toute entorse à ces consignes est pénalisée.
- Répondez aux questions sur la copie.
- En plus des règles usuelles des contrôles, il est strictement interdit de :
  1. commencer le contrôle avant le signal de démarrage de l'enseignant,
  2. continuer le contrôle après le signal d'arrêt de l'enseignant.
- Le barême est donné à titre indicatif.

Nom :
Prénom :
Groupe :

1 QCM - 5 points

Consignes : Entourez les numéros des propositions que vous estimez correctes. Pour chaque question, il se peut qu'aucune, qu'une ou plusieurs ou encore que toutes les réponses proposées soient correctes.
Principe de notation (à titre indicatif seulement) : 1 bonne réponse = 1 point, 1 réponse non fournie = 0 point, 1 mauvaise réponse = -1 point

1. En supposant que vous êtes l'utilisateur `bill`, quel affichage résulte des commandes suivantes ?

```
$ alias ma_racine="echo $HOME"
$ ma_racine="racine=$HOME"
$ echo "$(ma_racine)"
```

   (a) $HOME
   (b) /mci/ei1114/bill
   (c) racine=$HOME
   (d) echo /mci/ei1114/bill
   (e) racine=/mci/ei1114/bill

2. Vous avez créé un exécutable `mon_programme` dans le répertoire `~/CSC3002/TP23/`. Depuis votre répertoire de connexion, vous tapez la commande `mon_programme`. Que faut-il faire pour que cela fonctionne ?
   (a) Positionner les droits x pour vous sur l'exécutable.
   (b) Modifier la variable d'environnement LD_LIBRARY_PATH.
   (c) Modifier la variable d'environnement PATH.
   (d) Diffuser le chemin du répertoire dans lequel se trouve l'exécutable.

3. La commande `chmod 364 .` vient d'être exécutée. Entourez les réponses correctes.
   (a) `Le groupe peut lire les entrées du répertoire de travail.`
   (b) `L'utilisateur a le droit de créer un fichier dans le répertoire de travail.`
   (c) `L'utilisateur a le droit de lire le contenu du répertoire de travail.`
   (d) `Les autres peuvent lire les entrées du répertoire de travail.`
   (e) `Le groupe ne peut pas supprimer un fichier dans le répertoire de travail.`

4. Dans votre terminal, les lignes suivantes sont affichées :
```
$ ls monrapport.txt
ls: monrapport.txt: Aucun fichier ou répertoire de ce type
```

   Que se passe t-il ?

   (a) La commande `ls` affiche le contenu du fichier `monrapport.txt`.

   (b) La commande `ls` renvoie une erreur parce que le fichier de nom `monrapport.txt` n'existe pas dans le répertoire courant.

   (c) La commande `ls` n'existe pas.

   (d) La commande `ls` liste les informations liées à l'entrée `monrapport.txt` précisant que cette entrée n'est ni un fichier ni un répertoire.

5. Que fait la commande `grep fic1 a | wc -l` ?

   (a) Elle écrit dans le fichier `a` le nombre de lignes lues dans le fichier `wc`.

   (b) Elle n'affiche pas le nombre de lignes contenant la chaîne de caratères `a` dans le fichier `fic1`.

   (c) Elle compte le nombre de lignes du fichier `fic1` du répertoire `a`.

   (d) Elle écrit dans `a` les lignes où la chaîne de caractères `fic1` est lue et affiche leur nombre.

   (e) Elle affiche le nombre de lignes du fichier `a` contenant le motif `fic1`.

6. Pour visualiser le fichier texte de nom `"mon rapport prefere.txt"` se trouvant dans le répertoire courant, je tape la commande :

   (a) `more $(mon rapport prefere.txt)`

   (b) `less mon\ rapport\ prefere.txt`

   (c) `cat mon rapport prefere.txt`

   (d) `viz ./mon rapport prefere.txt`

   (e) `more 'mon rapport prefere.txt'`

2 Gestion d'inscriptions à des manifestations - 15 points

Consignes : Répondez à chaque question dans l'encadré associé. Si vous deviez ne pas avoir assez de place, signalez dans l'encadré que vous continuez à la fin de la copie.
Sauf mention contraire, on ne vous demande pas de traiter les cas où les paramètres des scripts sont incorrects.
Les questions sont indépendantes les unes des autres. Cependant, afin de répondre à une question donnée, vous devrez avoir lu les précedentes.
Vous avez à disposition les scripts `P.sh` et `V.sh` utilisés pendant le module.

Un organisateur souhaite avoir une estimation du nombre de participants aux manifestations qui ont lieu dans sa salle de spectacle. Pour cela, il demande aux participants de s'inscrire aux événements proposés.

1. **Création de l'événement :** Afin de recenser les inscriptions, l'organisateur doit initier chacun des événements. Pour cela, on souhaite écrire un script, nommé `init_evt.sh`, qui prend en paramètre le nom de l'événement à initialiser, crée un fichier vide selon le nom de l'événement suffixé par `.evt`. Si l'événement devait déjà exister, le script doit demander au préalable à l'utilisateur s'il convient de le réinitialiser. Dans le cas contraire, le script s'arrête.
   Une exécution possible est :

```
$ ./init_evt.sh Cyrano
Création de l'événement Cyrano.
$ cat Cyrano.evt
$ ./init_evt.sh Cyrano
L'événement Cyrano existe déjà. Souhaitez-vous le réinitialiser (o/n)?
n
$ cat Cyrano.evt
$ ./init_evt.sh Cyrano
L'événement Cyrano existe déjà. Souhaitez-vous le réinitialiser (o/n)?
o
Création de l'événement Cyrano.
$ cat Cyrano.evt
$ ./init_evt.sh Pirates_Caraibes
Création de l'événement Pirates_Caraibes.
$
```

   Écrivez le script `init_evt.sh` dans le cadre suivant :

2. **Inscription à l'événement :** Un participant s'inscrit à un événement en appelant un script nommé `inscription_evt.sh`. Ce script a trois paramètres : le nom de l'événement visé, le nom et l'âge du participant. Le script commence par vérifier si l'événement a été créé par le gérant, *i.e.* si le fichier associé à l'événement existe dans le répertoire courant. Doit ensuite être insérée dans le fichier une ligne constituée de plusieurs informations : le numéro d'inscription du participant suivi de son nom et de son âge. Pour déterminer son numéro d'inscription, vous pouvez utiliser le nombre de lignes du fichier.
   En supposant l'événement Cyrano créé, une exécution possible est :

```
$ ./inscription_evt.sh Cyrano Paul 19
Bravo Paul, votre participation à l'événement Cyrano est confirmée.
$ cat Cyrano.evt
1 Paul 19
$ ./inscription_evt.sh Cyrano Sidonie 12
Bravo Sidonie, votre participation à l'événement Cyrano est confirmée.
$ cat Cyrano.evt
1 Paul 19
2 Sidonie 12
$ ./inscription_evt.sh Tartuffe
Désolé, l'événement Tartuffe n'existe pas.
$ ./inscription_evt.sh Pirates_Caraibes Jack 34
Bravo Jack, votre participation à l'événement Pirates_Caraibes est
confirmée.
$
```

   Écrivez le script `inscription_evt.sh` dans le cadre suivant :

3. **Affichage de la liste des participants à l'événement :** L'organisateur souhaite pouvoir faire des statistiques sur ces visiteurs. On souhaite fournir un script, nommé `stat_evt.sh`, permettant de donner le nombre total de visiteurs attendus ainsi que le nombre de visiteurs mineurs. Il doit aussi afficher la liste des participants.
En supposant Paul et Sidonie inscrits à l'événement Cyrano, une exécution possible est :

```
$ ./stat_evt.sh Cyrano
Sur 2 participants attendus à l'événement Cyrano, 1 est mineur.
-------------
Liste des participants :
1 Paul 19
2 Sidonie 12
$ ./inscription_evt.sh Cyrano Joe 6
Bravo Joe, votre participation à l'événement Cyrano est confirmée.
$ ./stat_evt.sh Cyrano
Sur 3 participants attendus à l'événement Cyrano, 2 sont mineurs.
-------------
Liste des participants :
1 Paul 19
2 Sidonie 12
3 Joe 6
$
```

Écrivez le script `stat_evt.sh` dans le cadre suivant :

4. **Affichage en temps réel de la disponibilité de l'événement :** Afin de motiver ces visiteurs, l'organisateur souhaite afficher en temps réel les prévisions d'affluence des événements organisés sur son écran d'affichage. À cet effet, le script nommé `teaser_evt.sh` doit afficher toutes les secondes le nombre d'inscriptions à l'ensemble des événements disponibles. Dans une boucle infinie, le script doit dans l'ordre :
   — afficher le message d'accueil,
   — itérer sur les fichiers du répertoire courant suffixés par `.evt` ; sur chacun d'eux,
      — compter son nombre de ligne,
      — récuperer le nom de l'événement à partir du nom du fichier ; pour cela, vous pouvez utiliser la commande `cut` vue pendant le module qui permet de supprimer des parties de lignes d'un fichier passé en paramètre. Ici, vous devez supprimer un morceau du nom du fichier et non du contenu du fichier. Pour cela, utilisez la commande `cut` comme suit : `echo nom_fichier | cut -d "délimiteur" -f numero_champ_a_conserver`.
      — afficher le message adéquat,
   — dormir 1 seconde à l'aide de l'appel à la commande `sleep 1`.

   L'affichage doit prendre fin quand l'organisateur décide d'éteindre son écran, ce que nous matérialisons par l'envoi d'un signal SIGINT.
   Une exécution possible est :

```
$ ls
Abracadabra Cyrano.evt Pirates_Caraibes.evt Repertoire1 Repertoire2 Toto
$ ./teaser_evt.sh
Dans votre salle de spectacle préférée,
déjà 3 inscrits à Cyrano
déjà 1 inscrit à Pirates_Caraibes
Dans votre salle de spectacle préférée,
déjà 4 inscrits à Cyrano
déjà 1 inscrit à Pirates_Caraibes
Dans votre salle de spectacle préférée,
déjà 4 inscrits à Cyrano
déjà 10 inscrits à Pirates_Caraibes
^C
Shutdown
$
```

   Écrivez le script `stat_evt.sh` dans le cadre suivant :