CTP UOD2

Déroulement du CTP

Modalités de l'épreuve :

- Épreuve individuelle : aucun échange possible entre les étudiants
- Durée : 3 heures
- Tous documents autorisés
- Si vous utilisez du code tiers provenant d'internet, indiquez-le dans votre CR
- Pas de messagerie instantanée ni différée, d'échange de fichiers, pas de téléphone, etc.
- Toute tentative de fraude => conseil de discipline

A la fin de l'épreuve, vous déposerez sur moodle dans la zone de dépôt appropriée un compte rendu de votre travail, comportant :

- Les réponses aux questions posées, ainsi qu'un état d'avancement de votre travail (dans un fichier de réponses dont le nom est imposé)
- Tous les fichiers produits
- Le tout dans une ARCHIVE au format *.tar.gz

On suggère de lire tout le sujet avant de commencer de manière à anticiper les traitements à réaliser, certaines questions placées en fin du sujet pouvant avoir un impact sur la manière de réaliser les questions précédentes.

Consignes générales

Vous travaillerez chacun avec une machine possédant une distribution GNU/Linux.

Vous indiquerez pour chaque question les commandes que vous utilisez ainsi que la signification de ses options et toute autre information **pertinente**[1] montrant que vous savez ce que vous faites et êtes capables de mobiliser vos connaissances, de manière à aider le correcteur à "trouver des points".

A chaque fonctionnalité mise en œuvre, vous réalisez un jeu d'essai, de manière à constater que vos procédures fonctionnent, avant de passer à la fonctionnalité suivante. Vous rendrez compte de ces jeux d'essai, ainsi que des fichiers intermédiaires produits, avec des numéros de version, dans votre archive finale.

Travail préliminaire & travail final

Au début du CTP, ouvrez un interpréteur de commandes :

- Déclenchez l'enregistrement de votre session à l'aide de la commande ***script***.
  - Vous rappellerez son fonctionnement dans votre CR.
  - A la fin du CTP, vous terminerez la commande ***script***, pour générer une transcription de l'ensemble des opérations que vous avez réalisées dans votre terminal, et rendrez cette transcription au correcteur. Cela servira à l'évaluation.

1 i.e. en rapport avec l'énoncé...

- Créez un répertoire **CTP*NOM*_avril2019** en remplaçant ***NOM*** par votre nom.
  - Vous travaillerez dans ce répertoire pendant tout le CTP. Déplacez-vous dans ce répertoire.
  - Dans ce répertoire, créez un fichier **reponses*NOM*.txt** en remplaçant *NOM* par votre nom. Vous rédigerez vos réponses dans ce fichier.

A la fin de l'épreuve :

- Vous produirez une archive nommée **CTP*NOM*_avril2019.tar.gz** contenant tous les fichiers de votre répertoire **CTP*NOM*_avril2019** (y compris la transcription générée par la commande script, ainsi que le fichier de réponses) et vous déposerez cette archive sur moodle, dans la zone de dépôt prévue à cet effet du cours UOD.
- Indiquez dès maintenant dans votre fichier de réponses la commande qui vous servira à produire cette archive.

Bon travail à tous

Exercice 1 : Questions de Cours

1. Quel est la taille max d'un fichier dont le contenu est adressé par 3 blocs d'indirection de niveau 1,2 et 3 (blocs de 1024 octets, adresses sur 32 bits) ?
2. Quelle option de **ls** permet d'afficher le numéro d'inode d'un fichier ?
3. A quoi sert la variable d'environnement **PATH** ?
4. A quoi sert le bit suid ? Citer un programme avec son bit suid actif ?
5. Sur quelle classe d'utilisateurs faut-il positionner le sticky bit ?
   - Comment positionner (commandes absolues) un sticky bit et les droits rw pour tous (propriétaire, groupes et autres) ?
   - Comment positionner un bit suid (commande symbolique) ?
6. Que retourne une commande en cas de succès ?

Exercice 2 : Script Shell

Nous souhaitons ajouter à notre système un ensemble d'**applications ludiques** en ligne de commande, ainsi qu'un moyen d'accès unique pour les lancer, un système de gestion de high-scores et des outils de configuration permettant d'activer ou désactiver certains jeux proposés aux utilisateurs.

Partie 1 : architecture de configuration des jeux

Pour la configuration des jeux, l'architecture de notre système se présente sous la forme d'un répertoire **games-available** dans lequel se situent les jeux disponibles, et d'un autre répertoire **games-enabled** dans lequel se situent les jeux que l'on souhaite "activer". Pour activer un jeu, il faut qu'un lien symbolique soit présent dans le répertoire games-enabled et pointe vers un des jeux du répertoire games-available. Des scripts shell baptisés ***ser_eng <nom>*** ("enable game") et ***ser_disg <nom>*** ("disable game") permettent respectivement de créer ces liens. Cette architecture est directement inspirée de la configuration de serveur Web apache2.

1. Créez dans **CTPNOM_avril2019** le répertoire **ser_games** devant contenir les sous-répertoires **libsh**, **bin**, **data**, **games_available** et **games_enabled**.
2. Copiez dans le répertoire **libsh** les librairies de fonction shell créés lors des TPs précédents, et dans **games_available** le script shell correspondant au quizz développé en séance (ce sera notre premier jeu). Si besoin, modifiez le fichier quizz pour qu'il puisse faire appel aux librairies copiées dans libsh. Vérifiez que ce premier fonctionne en l'appelant directement depuis le répertoire **games_available**.
3. Créez dans **bin** les scripts ***ser_eng*** *et* ***ser_disg***. Chaque script devra :
   - Lire un argument en ligne de commande, devant correspondre au nom complet de l'un des fichiers du répertoire **games_available**
   - Vérifier la présence de ce fichier dans le répertoire **games_available**
   - Créer (***ser_eng***) ou supprimer (***ser_disg***) un lien vers ce fichier dans le répertoire **games_enabled**
4. Testez les scripts pour le jeu du quizz développé en séance : un lien vers le script de quizz doit apparaître et disparaître dans le répertoire **games_enabled**

Partie 2 : création d'un jeu simple : "multiplications"

5. Créez un second script de jeu dans le répertoire **games_enabled**, nommé "***multiplications***". Ce script devra :
   - Utiliser des fonctions de la librairie libsh créée plus haut
   - Tirer au sort deux entiers entre 1 et 10 : $E1 et $E2
   - Demander à l'utilisateur d'entrer le résultat du calcul du produit $E1 * $E2
   - Vérifier que le résultat est correct et l'afficher sur la sortie standard, puis se terminer

Partie 3 : lancement des jeux

6. Créez dans le répertoire **bin** un script ***ser_games*** qui permettra de lancer les jeux actifs. Ce script devra :
    - Demander à l'utilisateur son pseudo, et l'enregistrer dans une variable globale **$PSEUDO**.
    - Lister les jeux *actifs* en les préfixant par un numéro d'**inode** (i-number)
    - Demander à l'utilisateur de choisir le numéro du jeu auquel il souhaite jouer
    - Exécuter le jeu sélectionné
    - Lorsque le jeu est terminé, devant à l'utilisateur s'il veut jouer à un autre jeu ou terminer

Partie 4 : gestion des scores

7. Modifiez le script de quizz pour qu'il ajoute lorsqu'il se termine dans le fichier dénoté par la variable d'environnement **$SER_SCORE** le score du joueur, associé à son pseudo. Si votre script de quizz ne gérait pas de score, faites lui fabriquer un fichier score comportant un score aléatoire entre 1 et 20.
8. Modifiez le script de multiplication pour qu'il propose à l'utilisateur de poursuivre les calculs, et compte les points. Lorsque l'utilisateur décide d'arrêter, le script doit ajouter le score du joueur dans le fichier **$SER_SCORE**.
9. Après chaque ajout d'un score au fichier de scores, nous souhaitons réordonner ce fichier par ordre de scores décroissant. Mettez en oeuvre les fonctionnalités d'ajout d'un score et de tri du fichier de scores dans des fonctions de la librairie ***libsh*** et utilisez ces fonctions dans le script ***ser_games***.
10. Ajouter au menu du script ***ser_games*** la possibilité d'afficher les scores des 10 meilleurs joueurs.
11. Ajouter au menu du script ***ser_games*** la possibilité d'effacer tous les scores ou seulements ceux d'un certain jeu.

Partie 5 : méta-données des jeux

Nous souhaitons pouvoir associer aux jeux des méta-données (un nom complet, une description, des arguments par défaut à passer au jeu lors de son lancement) stockées dans un fichier nommé ***ser_data***

12. Proposez une structure de données pour le fichier ***ser_data*** et définissez les méta-données associées aux jeux "quizz" et "multiplication"
13. Ajoutez au menu du script ***ser_games*** une option permettant d'afficher les méta-données d'un jeu
14. Créez un script ***ser_config*** permettant de faciliter la saisie des méta-données d'un jeu, pour éviter de devoir éditer (création, modification, suppression des méta-données d'un jeu) le fichier ***ser_data*** manuellement.

Partie 6 : jeux supplémentaires (BONUS)

15. Proposez d'autres jeux faciles à développer à l'aide de scripts shells et expliquer comment vous vous y prendriez, sans les développer.