Raport

Réponse aux questions

- Partie 1 : files.sh
- Partie 2 : processes.sh
- Partie 3 : services.sh
- Partie 4 : main.sh
- Partie 5 : delegate.sh

  Chaque script contient un commentaire en début de chaque fonction indiquant le numéro de la question à laquelle elle répond.

Partie 1 : Fichiers

Menu principal Vu le nombre de fonctions différentes qu'on a implémenté, lorsqu'on affichait toutes les entrées possibles dans le menu principal, et quand on attendait que l'utilisateur choisisse un numéro pour éxécuter une des fonctions, le nombre de lignes de code était énorme. On a donc décidé d'implémenter une méthode générique permettant d'afficher le menu et d'éxécuter une des fonctions grâce à deux constantes `MENU_OPTIONS` (étant la description de chaque méthode) et `MENU_ACTIONS` (étant le nom de la méthode à éxécuter) qui sont des tableaux qui permettent de réduire drastiquement le nombre de ligne de code.

> Cette implémentation est utilisée dans les autres scripts grâce au fichier `common.sh` comportant les différentes méthodes pour faire l'affichage du menu utilisable.

Filtres La consigne sur l'implémentation d'un système de filtre n'étant pas très détaillé, nous avons dû en faire notre propre interprétation. Nous avons choisi un menu permettant de sélectionner les filtres souhaités de manière intéractive. La grande flexibilité de la commande find nous a permis de traduire les entrées de l'utilisateur en commande `find` facilement.

Difficultés Le gros du travail a été de proposer une interface console interactive à l'utilisateur. Il a été plus complexe d'implémenter proprement une interface compréhensible que de traduire les entrées de l'utilisateur en commande `find`.

Partie 2 : Processus

Filtres Contrairement à la partie 1, nous avons choisi de ne pas implémenter un système de filtre interactif. Nous avons préféré implémenter un système de filtre par argument. Cela nous a permis de simplifier le code et de rendre l'interface plus simple. Puisque nous passons par un menu principal, il n'est pas posssible de passer directement des arguments à la commande. Nous avons donc

créé une fonction intermédiaire présentant les options disponibles en proposant à l'utilisateur d'entrer les options qui seront ensuite passsé en argument à la fonction de filtrage.

Difficultés La commande `ps` est bien plus limité que `find` en terme de possibilité de filtrage. Tout les processus de filtrages sont donc effectués sur la sortie de la commande `ps` à l'aide des commandes `grep` et `awk`.

Limitations Il n'y a pas de possibilité de choisir pour chaque filtre si il doit s'agir d'un filtre ET ou OU.

Partie 3 : Services

Aucune difficulté particulière n'a été rencontrée lors de la réalisation de cette partie.

Partie 4 : Menu principal

Aucune difficulté particulière n'a été rencontrée lors de la réalisation de cette partie.

Partie 5 : Délégation

Il a fallu imaginer une manière de donner uniquement des droits d'execution à certains utilisateurs. Nous avons pour cela utilisé les groupes. Le script `delegate.sh` permet de créer un groupe `livl-bash` et d'y ajouter un utilisateur. Une fois ce groupe créé et l'utilisateur ajouté, il est possible de donner les droits d'execution au script `main.sh` à ce groupe.

Général

Fonctions communes Nous avons choisi de créer un fichier nommé `common.sh` qui contient des fonctions utilisées par plusieurs scripts. Il permet de réduire la duplication de code et de rendre le code plus lisible. Par exemple, il y avait une énorme redondance dans chaque fonction de chaque script quand on demandait à l'utilisateur de saisir un nombre / un certain format de date / ou une chaîne de caractères. La difficulté rencontré à été le fait qu'il soit impossible de retourner un `string` comme valeur de retour d'une fonction, donc la méthode a été de faire un `echo` de la chaine de caractère et de la récupérer dans une variable dans le fonction appelante.

Affichage Nous avons aussi implémenté une méthode permettant de rafraichir l'affichage de la console après qu'une fonction du menu choisis par l'utilisateur se termine. On a rajouté un timer qui décrémente chaque seconde pour une durée de 30 secondes et qui l'affiche à chaque fois à l'écran. Lorsque le timer

arrive à 0, la console est rafraichie et la fonction sera re-appelée. Si l'utilisateur appuie sur une touche, cela le redirige vers le menu principal.

La difficulté rencontrée a été d'implémenter une solution de rafraichir la console chaque seconde et que l'utilisateur puisse appuyer sur une touche, sans que cela ne soit bloquant. La solution a été de faire une tâche en arrière plan qui s'occupe de rafraichir la console afin qu'on puisse continuer à lire les entrées de l'utilisateur.

Style de la console Nous avons opté à une solution de style de console qui permet de la rendre plus lisible et pour rendre une utilisation plus agréable pour l'utilisateur. Nous avons utilisé des couleurs et des caractères spéciaux pour rendre le menu plus lisible.

Conclusion

Nous avons épluchés énormément d'aspects de bash. Néanmoins nous nous sommes rendu compte à certains moments que faire des scripts en bash présente des limites (comme par exemple, le fait qu'il soit impossible de retourner une valeur autre qu'un nombre dans une fonction bash).