Liste des sujets

2023/2024

Liste des projets d'applications du module "Système et scripting"

Sommaire

Préambule

Consignes

- L'application doit être modulaire et contenant des fonctions.
- Chaque option sélectionnée doit déclencher une fonction.
- L'utilisation de **getopts** est obligatoire pour gérer les options du script
- Le nom du module/script principal doit porter le nom l'étudiant.
- Si les commandes évoqués dans les sujets suivants sont introuvables via le shell, l'étudiant est amené à les installer via la commande apt ou yum.
- Toute modification des sujets doit de faire après confirmation de l'enseignant.
- Tout tentative de fraude sera pénalisée avec une note de mini projet égale à zéro.

Grille de validation

Description	Points
Implémentation du script	10
Réponse aux questions	5
gestion des erreurs	2
script modulaire	2
lisibilité: (commentaires, indentations, noms variables...)	1

Projet 1: Sauvegarde

Objectif

On se propose de créer un script **sauvegarde.sh** permettant de faire l'archive des fichiers de votre répertoire personnel.

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " **sauvegarde.sh**: [-h] [-g] [-m] [-v] [-n] [-r] [-a] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.
4. Ecrire une fonction qui permet d'afficher le nombre de fichier et la taille totale occupée des fichiers modifiés dans les dernières 24heures.
5. Ecrire une fonction qui permet d'archiver dans une « archive tar » (fichier *.tar.gz) tous les fichiers de votre répertoire personnel (/home/votre-nom) qui ont été modifiés dans les dernières 24 heures

Options

- -n : Pour afficher le nombre de fichier et la taille totale occupée des fichiers modifiés dans les dernières 24heures.
- -a : Pour archiver dans une « archive tar » (fichier *.tar.gz) tous les fichiers de votre répertoire personnel (/home/votre-nom) qui ont été modifiés dans les dernières 24 heures.
- -r : pour renommer l'archive avec la date et l'heure de la modification.
- -s FICHIER pour sauvegarder les informations sur les fichier archivé (nom-type-droit d'accès-date et heure de modification) dans un fichier passé en argument.
- -m : pour afficher un menu textuel (en boucle) qui permet d'accédé à chaque fonction
- -g : une fonction qui permet d'afficher un menu graphique avec plusieurs interface (vous pouvez utiliser YAD)
- .-v: Pour afficher le nom des auteurs et version du code.

- -h: Pour afficher le help détaillé à partir d’un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).

- Projet 2: Gestion de l'espace disque

Objectif

On se propose de créer un script disque.sh qui permet de lister, un par un, tous les fichiers faisant plus de 100 Ko dans l'arborescence du répertoire /home/utilisateur. Donnez à l'utilisateur la possibilité de supprimer ou de compresser le fichier.

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " disque.sh: [-h] [-j] [-s] [-p] [-l] [-v] [-m] [-g] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'un fichier texte contenant une description bien détaillée de l'application.
4. Une fonction qui permet de lister, un par un, tous les fichiers faisant plus de 100 Ko dans l'arborescence du répertoire /home/utilisateur.
5. Une fonction qui permet de supprimer un fichier si sa taille dépasse 100 Ko
6. Une fonction qui permet de compresser un fichier si sa taille dépasse 100 Ko.
7. Une fonction qui prend en arguments plusieurs fichiers de taille supérieur à 100 Ko et donne à l'utilisateur la possibilité de supprimer ou de compresser.
8. Une fonction qui permet d'écrire un journal avec le nom de tous les fichiers supprimés et l'heure de leur suppression.

Options :

- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalités de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et la version du code.
- -j : créer le fichier journal des fichiers supprimés
- -c : compresser un fichier
- -s : supprimer un fichier
- -p : parcourir des fichiers pour supprimer ou compresser
- -l : lister des fichiers de taille supérieur à 100ko
- -m : pour afficher un menu textuel (en boucle) qui permet d'accéder à chaque fonction

- Projet 3: Suppression sécurisée

Objectif

Implémenter, sous la forme d'un script, une commande « sécurisée » de suppression, sdel.sh. Les noms de fichiers passés en tant qu'arguments de la ligne de commande à ce script ne sont pas supprimés, mais, à la place, compressés avec gzip s'ils ne sont pas déjà compressés (utilisez file pour le vérifier), puis déplacés dans le répertoire ~/TRASH. Une fois lancé, le script vérifie si le répertoire ~/TRASH contient des fichiers datant de plus de 48 heures et les supprime de façon permanente.

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " sdel.sh: [-h] [-m] [-t] [-c] [-g] [-d] [-m] [-v] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.
4. une fonction qui permet de vérifier si un fichier passé en argument est compressé ou non.
5. une fonction qui permet de compresser avec gzip les fichiers passés en argument qui ne sont pas compressés.
6. une fonction qui permet de déplacer les fichiers compressés dan le répertoire /TRASH.
7. une fonction qui permet d'une part de vérifier si le répertoire /TRASH contient des fichiers datant de plus de 48 heures d'autres part de les supprimer de façon permanente.

Options

- -c : compresser
- -d : déplacer
- -t : supprimer ancien fichier
- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m : pour afficher un menu textuel (en boucle) qui permet d'accédé à chaque fonction

- Projet 4: Analyse du fichier /etc/passwd

Objectif

Implémenter un script analyse.sh qui permet d' analyser /etc/passwd et affichez son contenu sous forme plus lisible.

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " analyse.sh: [-h] [-v] [-t] [-m] [-g] [-v] [-n] [-p] user.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'un fichier texte contenant une description bien détaillée de l'application.
4. Une fonction qui permet de retourner le nombre d'utilisateurs
5. Une fonction qui affiche la liste des utilisateurs
6. Une fonction qui affiche en deux colonnes les noms des utilisateurs suivis par leurs répertoires personnels.
7. Une fonction qui liste les utilisateurs qui ont comme groupe primaire celui passé en argument.
8. Une fonction qui affiche le type de l'utilisateur (admin, applicatif, simple)

Options

- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel
- -n : afficher le nombre d'utilisateurs
- -p : afficher les répertoires personnels des utilisateurs
- -G : utilisateurs appartenant au même groupe
- -t : type de l'utilisateurs

Projet 5: Analyse du fichier /var/log/messages

Objectif

Implémenter un script log.sh qui permet d'analyser /var/log/messages pour produire un fichier joliment formaté des connexions des utilisateurs avec l'heure de connexion. Le script peut devoir se lancer en tant que root (conseil : cherchez la chaîne « LOGIN »).

Fonctionnalités

On se propose d'écrire un script secure.sh qui permet d'analyser des informations sur les fichiers logs

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " log.sh: [-h] [-m] [-g] [-c] [-l] [-a] [-b] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'un fichier texte contenant une description bien détaillée de l'application.
4. Écrire la fonction Connexion qui permet de tracer les tentatives de connexion de chaque utilisateur passé en argument dans un fichier nommer par le nom d'utilisateur. (extraire des lignes de fichier /var/log/secure)
5. Ecrire la fonction alert qui permet de tracer le nombre de mauvaise tentative de connexion de chaque utilisateur passé en argument (authentication failure) si le nombre dépasse 3 un message d'alerte sera afficher (analyse de fichier /var/log/secure)
6. Ecrire la fonction mail qui permet de tracer la date de dernier accès au service mail (/var/log/maillog)
7. Ecrire la fonction service qui permet d'analyser le fichier (/var/log/boot.log) et d'afficher les services qui sont entrain de démarrer (Starting nameservice)

Options :

- -c : tentatives de connexions
- -a : alert

- -l : mail
- -b : starting name service
- -h: Pour afficher le help détaillé à partir d’un fichier texte
- -g: Pour afficher un menu graphique (Utilisation de YAD).
- -m: Pour afficher un menu textuel

Projet 6: Surveiller un utilisateur

Objectif

Implémenter un script surveiller.sh qui surveille et trace automatiquement l'activité de la semaine d'un utilisateur et supprime les entrées qui datent de plus de sept jours.

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " surveiller.sh: [-h] [-d] [-s] [-j] [-v] [-g] [-m] user.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.
4. Une fonction qui permet d'afficher les dernières connexions d'un utilisateur
5. Une fonction qui crée un fichier journal contenant les connexions de l'utilisateur
6. Une fonction qui supprime les entrées qui datent de plus de sept jours du fichier journal.

Options

- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel
- -d : dernières connexions
- -s : supprimer
- -j : journaliser

Projet 7: Tester des mots de passe

Objectif

On se propose d'écrire un script password.sh qui permet de vérifier et valider les mots de passe. Le but est de marquer les candidats « faibles » ou facilement devinables.

- Un mot de passe en test sera en entrée du script via un paramètre de la ligne de commande.
- Pour être considéré comme acceptable, un mot de passe doit satisfaire les qualifications minimums suivantes :
  - Longueur minimum de huit caractères
  - Contenir au moins un caractère numérique
  - Contenir au moins un caractère non alphabétique, numérique et figurant : @, #, $, %, &, *, +, -, =
- Faire une vérification de type dictionnaire sur chaque séquence d'au moins quatre caractèresconsécutifs du mot de passe en test. Ceci éliminera les mots de passe contenant des « mots » disponibles dans un dictionnaire standard.

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " password.sh: [-h] [-v] [-m] [-g] [-N] [-t] mot.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.

Options

- -t : vérification du mot de passe introduit
- -h: Pour afficher le help détaillé à partir d'un fichier texte

- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

.

Projet 8: Journal des accès aux fichiers

Objectif

Ecrire un script trace.sh qui permet de tracer tous les accès aux fichiers dans /etc sur une journée. Ce journal devra inclure le nom du fichier, le nom de l'utilisateur et l'heure d'accès. Vous indiquerez si le fichier a été modifié. Écrivez toute cette information dans un fichier journal, sous forme d'une suite d'enregistrements clairement séparés par des tablutations.

Fonctionnalités

On se propose d'écrire un script journal.sh qui permet d'afficher des informations sur des dossiers donnés en paramètre

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " trace.sh: [-h] [-T] [-t] [-n] [-N] [-d] [-m] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.
4. Ecrire la fonction **AffricheFile** qui permet de tracer le nom de dossier ainsi que les noms des fichiers de ce dernier dans un fichier qui a pour nom nomdefichier_journal**.(cut –d …)**
5. Ecrire la fonction **AffricheDirectory** qui permet de tracer le nom de dossier ainsi que les noms des dossiers de ce dernier dans un fichier qui a pour nom nomdedossier_journal. **(cut –d ..)**
6. Ecrire la fonction **NB** qui permet de tracer le nombre des dossiers et des fichiers de cet argument dans un fichier qui a pour nom **count**. **(wc)**
7. Ecrire la fonction **DirectoryUser** qui prend en argument un dossier et qui permet de tracer le nom de son propriétaire à l'entête **(ls, cut)**
8. Ecrire la fonction **dateAcess** qui permet d'écrire dans le fichier date_journal la date de son dernier accès. (**stat file)**
9. Ecrire la fonction **datemodif** qui permet d'écrire dans le fichier modif_journal la date de sa dernière modification.(**stat file)**
10. Ecrire la fonction **stat** qui permet d'Afficher les statistiques qui permettent de visualiser un Dashboard qui a le rôle d'indiquer le nombre de fichiers et le nombre de dossiers dans ce dossier **(plot).**

Options

- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

Projet 9: Renommer les fichiers

Objectif

Implémenter un script rename.sh qui permet de renommer les fichiers.

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " rename.sh: [-h|--help] [-T] [-t] [-n] [-N] [-d] [-m] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.
4. Ecrire une fonction qui change tous les noms de fichier ou du répertoire passé en paramètre en minuscule.
5. Ecrire une fonction qui renommer les fichiers dont le nom contient des espaces.
6. Ecrire une fonction qui permet d'enlever l'extension du nom du fichier
7. Ecrire une fonction qui change tous les noms de fichier ou du répertoire passé en paramètre en majuscule.
8. Ecrire une fonction qui ajoute _d aux noms des repertoires passés en paramètre
9. Ecrire une fonction qui prend en argument le nom d'un fichier et une extension. La fonction renomme le fichier en ajoutant l'extension.

Options

- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

Projet 10: Informations sur les répertoires

Objectif

On se propose d'écrire un script inforep.sh qui permet d'afficher des informations sur des arguments donnés en paramètre

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " inforep.sh: [-h|--help] [-T] [-t] [-n] [-N] [-d] [-m] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.
4. Si l'argument est un dossier :

   4.1 Ecrire la fonction AfficheNbF qui permet d'Afficher le nombre de fichiers de cet argument (ls -l grep ^d wc –l)
   4.2 Ecrire la fonction AfficheNbD qui permet d'Afficher le nombre de dossiers dans cet argument(grep,wc)
   4.3 Ecrire la fonction TypeFiles qui permet d'Afficher les types de fichiers (cut-d)
   4.4 Ecrire la fonction AcessFiles qui permet d'Afficher les noms des fichiers ainsi que leurs droits d'accès. (cut)
   4.5 Ecrire la fonction PropFiles qui permet d'Afficher les noms des fichiers ainsi que leurs propriétaires. (cut)
   4.6 Ecrire la fonction stat qui permet d'Afficher les statistiques qui permettent de visualiser un Dashboard qui a le rôle d'indiquer le nombre de fichiers et le nombre de dossiers (plot)

5. Si l'argument est un fichier :

   5.1 Ecrire la fonction TypeFile qui permet d'Afficher le type de ce fichier
   5.2 Ecrire la fonction AcessFile qui permet d'Afficher ses droits d'accès. (cut)
   5.3 Ecrire la fonction PropFile qui permet d'Afficher le group et le propriétaire de fichier.
   5.4 Ecrire la fonction Ecriture qui permet d'Afficher les utilisateurs qui ont le droit d'écriture de ce fichier

Options :

- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

Projet 11: Gestion des utilisateurs

Objectif

Implémenter un script gestion_user.sh qui permet de gérer les utilisateurs.

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " **gestion_user.sh**: [-h|--help] [-T] [-t] [-n] [-N] [-d] [-m] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.
4. verouiller le compte d'un utilisateur et l'ajouter un fichier liste_user_v
5. déverouiller le compte d'un utilisateur et le supprimer du fichier liste_user_v
6. modifier le repertoire personnel de l'utilisateur et copier le contenu de son ancien répertoire vers le nouveau

Options

- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

Projet 12 : Gestions des favoris

Objectif

On se propose de créer un script sauv_favori permettant de sauvegarder un nouveau favori

Fonctionnalités :

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " sauv_favori: [-h|--help] [-T] [-t] [-n] [-N] [-d] [-m] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.

Options

- -a : Cette option évoque une fonction qui prend un argument (une chaine sans espace) et ajoute le chemin (après vérification de son existence) dans votre liste de favoris .
- NB : liste de favoris (un fichier qui contient des lignes. Chaque ligne est un chemin absolu d'un répertoire ou fichier.
- -c : Cette option déclenche une fonction qui se déplace dans un répertoire favori. Cette fonction prend un argument (une chaine sans espace) et cherche dans la liste favoris. Si le favori existe, la fonction change le répertoire de travail, sinon, elle ne fait rien.
- -r : Supprimer un favori de la liste
- -l : afficher la liste de tous les favoris avec l'option
- -h : l'option -h pour afficher le help détaillé à partir d'un fichier texte
- -g : pour afficher un menu graphique (exemple via YAD)
- -m : pour afficher un menu textuel (menu en boucle)
- -s : sauvegarder des images passé en argument par l'option -s dans un
- Répertoire favori_images, vérifier leurs nombres de pixels et les fixer à une valeur au choix si elles dépassent 700x700.

- -n : Pour renommer les images sauvegardés en adoptant ce format NOM_DATE_HEURE.jpg
- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

Projet 13: Statistiques sur les fichiers

Objectif

On se propose de créer un script statistic_file permettant d'avoir des statistiques sur les fichiers.

Fonctionnalités :

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " statistic_file: [-h|--help] [-T] [-t] [-n] [-N] [-d] [-m] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.
4. Le script devra fournir une fonction qui analyse le contenu du répertoire courant et de ses sous-répertoires et afficher des statistiques.

Options

- -T : Taille totale occupée (en Mo/Ko/Go)
- -t : Taille occupée par les fichiers/répertoires cachés
- -N : nombre de fichiers/répertoires(non vides)/ liens symboliques .
- -n : affiche le Nombres de petits fichiers (moins de 512ko) et gros fichiers (plus de 15 Mo)
- -s : affiche le Nombre de fichiers et répertoires vides (qui peuvent probablement être supprimés)
- -d : Nombre de fichiers Python, ou html, etc…
- -h : Pour afficher le help détaillé à partir d'un fichier texte
- -m : pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).

[redacted]

[redacted]

- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

Projet 14: System-monitor

Objectif

On se propose de créer un script system-monitor qui effectue un mécanisme de surveillance de plusieurs éléments :

v. l'activité CPU du système d'exploitation ;

w. l'utilisation de la mémoire du système d'exploitation ;

x. le contrôle de processus, par le nombre de processus actifs ;

Fonctionnalités

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " system-monitor: [-h|--help] [-T] [-t] [-n] [-N] [-d] [-m] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant une description bien détaillée de l'application.

.

Options

- -m : appelle une fonction qui affiche les quantités totales de mémoire et de zone de swap libres et utilisées (pour ceci vous utilisez la commande free : voir rappel) et stocke le résultat dans un fichier de trace /var/log/surveillance.log an ajoutant la date et l'heure de l'affichage.
- -c : appelle une fonction pour surveiller la charge d'entrée/sortie des périphérique du système depuis le dernier démarrage avec 3 intervalles de 5 secondes (pour ceci vous utilisez la commande iostat : voir rappel). Cette fonction stocke le résultat dans un fichier de trace /var/log/surveillance.log an ajoutant la date et l'heure de l'affichage.
- -p : appelle une fonction qui affiche une liste des processus en cours d'exécution formatée avec les colonnes (USER,PID ,PPID,numéro du processus, %mémoire , %cpu,COMMAND) triés par ordre décroissant selon l'occupation mémoire et cpu , avec

le nombre total des processus actifs (utilisation de la commande ps) . Le résultat sera stocké dans un fichier de trace /var/log/surveillance.log an ajoutant la date et l'heure de l'affichage.

- -l : appelle une fonction pour lister les messages de surveillance à partie du fichier de trace /var/log/surveillance (affichage inversé page par page).
- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

Rappel :

Commande free : pour afficher l'utilisation de la mémoire (Pour détails voir man free)

Commande ps : pour afficher l'état des processus (Pour détails voir man ps)

Commande iostat : pour afficher les statistiques de l'activité CPU (Pour détails voir man iostat).

Projet 15 : Compiler des fichiers en C.

Objectifs

On se propose d´écrire un script compiler.sh qui servira à compiler des fichiers C.

Fonctionnalités :

1. Ecrire la fonction show_usage qui affiche sur la sortie standard le message

   " compiler.sh : [-h|--help] [-T] [-t] [-n] [-N] [-d] [-m] [-s] chemin.." .

2. Le script doit tester la présence d'au moins un argument, sinon il affiche l'usage sur la sortie d'erreur et échoue.
3. fonction nommée HELP qui permet d'afficher le help à partir d'n fichier texte contenant

les différentes utilisations du script, qui sont :

compiler.bash -h|--help

compiler.bash --touch|--clean fichier.c ...

compiler.bash [option ...] --cc fichier.c ... où option est --debug|--optim|--warni

compiler.bash -g

Options

- –touch : fonction qui reçoit des noms de fichiers C en argument. Pour chaque fichier, on vérifie son existence, le met à la date courante et affiche le nom du fichier (sans extension) avec sa date de modification.
- --clean : fonction qui reçoit des noms de fichiers C en argument. Pour chaque fichier, on vérifie son existence, supprime le nom de l'exécutable correspondant.
- --cc : fonction qui reçoit des noms de fichiers C en argument. Pour chaque fichier, on vérifie son existence, puis compile le fichier C avec gcc, en lui fournissant selon les flags les options nécessaires (-g ou -O2 ou -Wall -W, voir rappels).
- --debug : évoquer l'option –g de gcc
- --warni : évoquer l'option -W de gcc
- -g : activer un menu graphique

- -h: Pour afficher le help détaillé à partir d'un fichier texte
- -g: Pour afficher un menu textuel et gérer les fonctionnalité de façon graphique(Utilisation de YAD).
- -v: Pour afficher le nom des auteurs et version du code.
- -m: Pour afficher un menu textuel

Rappels

La commande gcc [option ...] fich.c -o fich compile un fichier fich.c en un exécutable fich. Si l'option -g est présente, gcc rajoute des informations de débogage dans l'exécutable, qui seront utiles pour le débogueur gdb. Si l'option -O2 est présente, gcc optimise le code produit. Si les option -Wall -W sont présentes, gcc affiche des warnings.