PROJET SYSTEME

Le Tp a été réalisé par le trinome :

- Ait benali Faycal (g02)
- Fodil Zine-eddine
- Aiche Mohammed Islam

Introduction:

Le principal du projet est de faire tourner un shell permettant de traiter les tarballs comme il s'agissait de répertoires, sans que les tarballs ne soient désarchivés. le shell aur

C'est quoi un shell :

Le shell est un programme qui assure l’interface entre l’utilisateur et le système Unix. ou bien, il permet à l'utilisateur d'interagir avec le systeme via un terminal. Il est aussi appelé un interpréteur de commandes. il possède deux modes d'utilisation : -interactif : l'utilisateur saisit et exécute ses lignes de commande une a une dans un terminal. -non interactif : le shell lit un ensemble de commande dans un fichier appelé shell script.

caractéristiques :

Les interpretteurs de commande ont les caractéristiques communes suivantes : * permet a l'utilisateur de lancer des commandes. * Ils possèdent des commandes internes et des mots-clés(faire de la programmation) Chaque shell propose ses propres caractères spéciaux, commandes internes... Mais les interpréteurs les plus utilisés ont un certain nombre de fonctionnalités en commun. You can also:

- Import and save files from GitHub, Dropbox, Google Drive and One Drive
- Drag and drop markdown and HTML files into Dillinger
- Export documents as Markdown, HTML and PDF

Fonctionnement du shell :

Statégie adoptée :

Au lancement du shell,une fenetre apparait permettant à l'utilisateur de saisir la commande souhaitée. La première phase de traitement est entamée dès que l'utilisateur valide la commande entrée, le programme commence alors à effectuer les traitements sur la ligne de commande en vérifiant d'abord si cette dernière est différente de la chaine vide ... dans le cas contraire, le programme se relance et redonne la main à l'utilisateur pour taper de nouveau la commande valide! Après avoir vérifié que la commande saisie n'est pas vide, elle sera sauvegardée dans l'historiquede commandes avant que la deuxième phase de traitement ne soit commencée. Dans La deuxième phase du traitement, la commande sera analysée,au début on verifie si elle ne correspond pas à la commande ("exit") qui engendre automatiquement l'arret du programme. Dans le cas contraire, la chaine de caractère correspondante à la commande est décortiquée. En effet, la ligne sera analysé du premier caractère au dernier pour en tirer le nom complet de la commande, les éventuels paramètres, les fichier ou les noms de repertoires entrés. en suite, une études de cas sera faite sur l'ensemble des paramères : inexistance,invalidité ... Dans le cas ou la commande rentrée n'est pas

valide, le programme affiche un message d'erreur est affiché à l'utilisateur, et le shell se remet à son état initial (l'utilisateur pouura a nouveau retaper la commande et le processus se répète). une fois l'analyse assure la validité de la commande , le shell lance l'éxécution de cette dernière, et la traite soit avec pipes, soit sans pipe.

Architecture logicielle :

" L’architecture logicielle décrit d’une manière symbolique et schématique les différents éléments d’un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. Contrairement aux spécifications produites par l’analyse fonctionnelle, le modèle d'architecture, produit lors de la phase de conception, ne décrit pas ce que doit réaliser un système informatique mais plutôt comment il doit être conçu de manière à répondre aux spécifications. L’analyse décrit le « quoi faire » alors que l’architecture décrit le « comment le faire » 1.
https://fr.wikipedia.org/wiki/Architecture_logicielle

Dans ce qui suit,nous allons présenter l'architecture logicielle de notre solution:

Diagramme cas d'utilisation :

Dans notre projet, il existe seulement trois cas d'utilisation qui sont :

- ouvrir shell : l'utiliisateur pourra ouvrir le shell en lancant le programme
- fermer shell : l'utilisateur ferme le shell en exécutant la commande exit.
- saisir commande : l'utilisateur est invité a saisir la commande souhaitée. La figure suivante représente le diagramme de cas d'utilisation du shell (en UML)

---le diagramme ici

Le découpage modulaire :

Dès que l'on écrit un programme de taille importante il est indispensable de se fixer un certain nombre de règles d'écriture. En particulier, il est nécessaire de fractionner le programme en plusieurs fichiers sources, que l'on compile séparemment... autrement dit, on applique la loi 'diviser pour reigner' dans ce qui suit nous proposons un découpage modulaire pour la solution

- BOOLEEN Commande_Vide(String *CMD)

  > *Cette fonction vérifie si la commande n'est pas vide Elle renvoie un booléen qui vaut 1 si la ligne est vide, 0 sinon*

- VOID Sauvegarder_Commande (String CMD)

  > *Cette procédure sauvegarde la commande dans l'historique de commandes Elle prend comme paramètre la commande et la sauvegarde dans une structure de données*

- BOOL Cmd_Exist (String CMD)

  > *Cette fonction vérifie dans la structure de données si la commande "CMD" existe Elle renvoie 1 si la commande existe , 0 sinon*

- STRING Extraire_Nom (String CMD)

  > *Cette procédure permet d'extraire le nom de la commande de la ligne decommande pour pouvoir vérifier ensuite si elle existe ou non La commande*

> *renvoit le nom de la commande dans un string*

- VOID Extraire_Parametres(String CMD, String Tableau[])

  > *La procedure extrait les paramètres de la ligne de commande et les met dans le tableau tableau[] qu'on lui donne comme parametre*

- BOOL Parametres_Valides(String CMD, Tableau[])

  > *La fonction vérifie la validité des paramètres La fonction renvoie 1 si les parametres de la commande CMD sont valides, 0 sinon*

- BOOL Chemin_Valide (String CMD)

  > *Cette fonction vérifie la validité du chemin Elle renvoie 1 si le chemain est valide, 0 sinon*

Les structures de données :

Dans tout le projet, on considère la commande comme enregistrement ayant les éléments suivants : TYPE CMD = Enregistrment

> *nom : chaine de caractères parametres : tableau de chaine de caracteres*

FIN

Pour sauvegarder les commandes, on aura besoin d'une structure de données :

TYPE Historique = Enregistrement

> *historique : tableau d'élément de type CMD*

FIN

On aura besoin d'une strcuture de données pour sauvegarder aussi la liste des commandes existantes qui sont au nombre de 10 (mentionnées dans l'énnoncé)

Type Liste_CMD = tableau[10] de CMD

Les algorithmes implémentés :

bool commande_vide(string cmd) debut

> *var vide =faux if (cmd == "") vide = vrai ; return vide;*

fin

String extraire_nom(string CMD) int main() { char cmd[]; int i=0; char str[] = "Hello welcome to our application"; int len = strlen(str); char d[] = " "; // définir un délimiteur qui est l'espace char *p = strtok(str, d); // renvoie un pointeur vers le prochaine caractere while(p != NULL) { cmd[i]=p; // on recupere un tableau ou les mots de la commande son séparés i++; p = strtok(NULL, d); }

return cmd[0]; // renvoie le premier element qui est le nom }

bool CMD_Existe(String cmd) {

> *string s= extraire_nom(cmd); trouv=faux for(int i=0;i<10;i++) { if (s== Liste_CMD[i] trouv=vrai) } return trouv;*

}