Editeur de projet : Elias Khnissi, Gabin Gaucher—Boissier

CréerTable.c :

Nous comptions faire un fichier .c qui quand on le lance exécute un fichier Bash qui avec les bon paramètres crée l’archive au nom demandé, après quelques instant de réflexion nous nous sommes dit que de toute façons nous allions utiliser la fonction execv().

C’est donc une piste que nous avons abandonné vite dans le développement car cela allais simplement complexifier la chose pour rien, en théorie c’est une façon de faire mais de notre point de vu pas la plus simple.

Gestion des erreurs :

Dans cette catégorie nous regroupons :

Une mauvaise utilisation de la commande.

Un problème d’exécution d’une fonction du programme

Dans notre programme la première erreur qui est possible c’est que les paramètres de la commande soit mauvais par exemple donner 2 paramètres alors que le programme en attend qu’un : le nom du la future archive.

Pour cela nous avons mis en place une condition qui affiche dans un premier temps une erreur et un rappel d’usage dans le terminal et dans un deuxième temps ferme le programme avec un code d’erreur.

Pour cette condition nous vérifions que la variable argc est égale à 2 (pourquoi pas 1) car de base quand le programme est lancé sans autre paramètres sur la ligne de commande argv est un tableau de chaine de caractère contenant uniquement le nom de la commande tapée donc argc à ce moment-là vaut 1. Notre programme doit avoir un et un seul paramètre donc argc doit être égale à 2 car argv contiendra le nom de la commande et l’unique paramètre que nous désirons.

Pour ce qui est des erreurs liées à l’exécution d’une fonction dans le programme, les possibilités sont multiples : une erreur peut survenir à l’exécution de la fonction fork ,mais aussi à l’exécution de la fonction execv.

Pour la fonction fork l’erreur est visible par la valeur de la variable mon\_pid de type pid\_t si cette valeur est négative alors c’est que le processus n’a pas été créé, dans le cas contraire la fonction à bien fonctionné.

La gestion d’une erreur avec la fonction execv est plutôt simple car execv quand elle est appelé la fonction coupe le programme courant par l’exécution de la commande demandé, donc si la fonction ne se fait pas le programme courant s’exécute toujours ce qui fait que la solution est d’écrire une instruction qui affiche proprement un message d’erreur comme la fonction fprintf avec le paramètre stderr pour afficher dans la sortie standard d’erreur et après cette appel de fonction de retourner un code d’erreur pour fermer le programme.

Il y a aussi une autre erreur à prendre en compte, le mauvais usage de la commande mkdir, pour l’instant le programme est assez robuste pour la gestion d’erreurs évidentes, comme le mauvais appel de la commande creerTable, des erreurs internes à cause des fonctions de C utilisés. Mais si nous donnions un paramètre comme demandé, mais que ce nom de table soit déjà existant, ce qui se passera c’est que la fonction execv va bien s’exécuter mais nous afficher un message d’erreur de la commande mkdir nous informant que le dossier existe déjà. Nous voulons que ce message en anglais n’ai pas l’occasion d’apparaitre.

Donc nous avons fait l’utilisation de la bibliothèque [sys/stat.h](https://www.man7.org/linux/man-pages/man0/sys_stat.h.0p.html) (le lien redirige vers le Man de cette bibliothèque),une autre source qui nous a permis d’écrire ce programme : [vérifier si un répertoire existe](https://shinyu.org/fr/c/files-and-io/checking-if-a-directory-exists/#:~:text=En%20C%2C%20l%E2%80%99existence%20d%E2%80%99un%20r%C3%A9pertoire%20peut%20%C3%AAtre%20v%C3%A9rifi%C3%A9e,si%20les%20informations%20r%C3%A9cup%C3%A9r%C3%A9es%20correspondent%20%C3%A0%20un%20r%C3%A9pertoire.), et un [man qui décrit la structure stat](http://manpagesfr.free.fr/man/man2/stat.2.html). Nous avons utilisé une fonction nommé: stat qui à pour prototype :

int stat(const char \*path, struct stat \*buf);

Cette function permet de réunir des informations liées à l’emplacement visé par le chemin d’accès passé en paramètre Path, toutes ces informations sont redirigées dans la structure stat qui doit être déclaré antérieurement mais indique aussi avec son return l’état du fichier/Dossier donné par le path (1 si il n’existe pas 0 si il existe). Dans la structure nous pouvons notamment avoir l’information : st\_mode qui nous est utile pour l’utilisation de la macro POSIX S\_ISDIR qui vérifie avec les donnés de st\_mode si effectivement les donnés de protection du fichier/dossier correspond à ceux d’un dossier return 0 si c’est un dossier et 1 si c’est autre chose qu’une archive.

Développement du programme :

Pour ne pas partir de rien nous avons récupéré le code que l’on avait fait en cours qui devait copier un répertoire dont on avait donné le nom en paramètre. Nous avons changé la commande utilisé dans le execv() par mkdir : commande qui crée une nouvelle archive avec le pour nom ce qu’on lui donne en paramètre.

l’exécution du execv nous lui avons donné une variable mes params qui contient le nom de la fonction que nous appelons : mkdir, le paramètre nom du nouveau dossier, et enfin un paramètre char\*=NULL pour annoncer la fin de la ligne de commande.

SupprimerTable.c

Pour supprimer la table nous partons de la même base avec les mêmes contrôles d’erreur, comme une mauvaise utilisation des paramètres que l’on doit fournir au programme quand on le lance, un chemin d’accès erroné/qui n’existe pas.