**Compte-rendu – Projet MiniShell**

Le mini Shell produit au cours de ce projet comporte les caractéristiques suivantes.

**1) Interprétation des commandes intégrées:**  
Ce MiniShell peut **exécuter des commandes qu’il intègre**.

Ces dernières sont:  
-**exit** et son alias **quit**, deux commandes permettant de **fermer le MiniShell**.  
-Les **commandes de gestion de jobs** décrites dans **g partie 8** de ce compte-rendu.

Si l’utilisateur demande **à exécuter une** (**série de**) **commande**(**s**) et **que cette dernière n’est composée que de commandes intégrées**:  
->**Le MiniShell ne génère pas de nouveau job**.

->Les **tests** associés à ces commandes (**exécutées de manière simple**, sans pipes ni redirections) sont décrits dans les fichiers **test\_exit.txt**, **test\_exit\_bis.txt** et **test\_commande\_integree\_ne\_cree\_pas\_de\_job.txt**.

**2) Interprétation de commandes non-intégrées:**  
Ce MiniShell peut **exécuter des commandes qu’il n’intègre pas**.

Ces commandes sont **celles décrites dans les dossiers pointés par la variable d’environnement PATH**.

**Chaque commande exécutée contenant une commande non-intégrée génère un job**. Ces derniers sont listés dans une **variable externe** (voir la **partie 8** pour plus de détails).

->Les **tests** associés à ces commandes (**exécutées de manière simple**, sans pipes ni redirections) sont décrits dans les fichiers **test\_commande.txt**, **test\_commande\_multiple.txt** et **test\_man.txt**.

**Gestion des erreurs de noms de commandes:**  
Si l’utilisateur **demande à exécuter une commande inconnue**, **un message d’erreur est affiché**.

->Les **tests** associés à ces erreurs (**exécutées de manière simple**, sans pipes ni redirections) sont décrits dans le fichier **test\_commande\_inexistante.txt**.

**3) Interprétation de commandes avec redirections d’entrée et/ou de sortie:**  
Ce MiniShell peut **exécuter des commandes avec redirection d’entrée et/ou de sortie**.

->Les **tests** associés à ces commandes (exécutées de manière simples, sans pipes ni redirections) sont décrits dans les fichiers **test\_redirection.txt**.

**Gestion des erreurs de redirection d’entrée/sortie:**  
Si l’utilisateur **indique une redirection vers un fichier n’existant pas**/**sur lequel il n’a pas les droit nécessaires pour effectuer les redirections** (par exemple si l’utilisateur indique une **redirection de la sortie vers un fichier auquel il n’a pas accès en écriture**), **un message d’erreur est affiché**.

Les **redirections** sont **possibles pour les commandes intégrées** et **non-intégrées**.

->Les **tests** associés à ces erreurs sont décrits dans les fichier **test\_erreur\_fichier\_entree.txt**, **test\_erreur\_fichier\_sortie.txt** et **test\_erreur\_fichier\_entree02.txt**.

**4) Interprétation d’une série de commandes reliées par des tubes:**  
Ce MiniShell peut **exécuter une série de commandes** dont les différentes parties **sont reliées par des tubes**.

Les **tubes** sont **possibles pour les commandes intégrées** et **non-intégrées**.

->Les **tests** associés à cette fonctionnalité sont décrits dans les fichiers **test\_pipe\_simple.txt**, **test\_pipe\_simple02.txt** et **test\_pipe\_multiple.txt**.

**5) Exécution de commandes en arrière-plan:**  
Ce MiniShell peut **exécuter une commande ou une série de commandes en arrière-plan**. C’est à dire que **durant l’exécution de la** (série de) **commande**(s)**, l’utilisateur pourra continuer à utiliser le MiniShell pour effectuer d’autres actions en parallèle**.

Une **variable externe** a été créee pour **garder en mémoire le numéro du job actuellement en exécution au premier plan**: ***numJobCommandeForeground***.  
->Si **aucun job n’est actuellement en exécution au premier plan**, ***numJobCommandeForeground* = -1**.

->Les **tests** associés à cette fonctionnalité sont décrits dans les fichiers **test\_execution\_background.txt**.

**6) Gestion des processus terminés:**  
**Lorsqu’un processus lancé par ce MiniShell se termine**, **le MiniShell l’intercepte** afin **d’éviter une invasion de processus zombis**.

**Le MiniShell ne rend la main à l’utilisateur suite au lancement d’une commande en foreground** que **lorsque tous les processus fils créés pour l’exécution de cette commande** (= lorsque tous les processus associés au jobs de cette commande) **se sont terminés et ont été interceptés**.

->Les **tests** associés à cette fonctionnalité sont décrits dans les fichiers **test\_gestion\_zombie.txt**.

**7) Gestion des processus de premier plan:**  
Lorsqu’un job est **exécuté au premier plan il peut être amené à changer d’état**,suite à la **réception d’un signal *SIGINT*** (**ctrl+C**)ou ***SIGTSTP*** (**ctrl+Z**).

Ces signaux sont donc **traités par le MiniShell**.

Lors de la **réception du signal SIGINT**, le **processus au premier plan est terminé et tué**. Lors de la **réception du signal SIGTSTP**, il est **mis en pause** et **placé en arrière-plan**.

->Les **tests** associés à cette fonctionnalité sont décrits dans les fichiers **test\_signal\_sigint.txt** et **test\_signal\_sigtstp.txt**.

**8) Gestion des jobs:**  
Tous les **processus directement lancés par un Shell** sont appelésdes **jobs**. Les **jobs associés à notre MiniShell** sont **listés dans une variable externe**: ***listeJobsShell***.

Cette liste est **accessible à l’utilisateur via la commande jobs**, qui **affiche sur le terminal** **la liste des jobs** avec **leur numéro**, **leur état actuel** (en cours d’exécution, arrêté, …)et **la commande ayant créé le job**.

**L’état des jobs peut être modifié** via les commandes **fg, bg et stop**:

* **fg** : **exécute au premier plan**, un processus qui **était en arrière-plan**.
* **bg** : **met en exécution** un processus **arrêté en arrière-plan**.
* **stop** : **arrête** un processus **en cours d’exécution en arrière-plan**.

->Les **tests** associés à cette fonctionnalité sont décrits dans les fichiers **test\_jobs\_fg.txt**, **test\_jobs\_bg.txt** et **test\_jobs\_stop.txt**.