**Rapport de Développement – MiniShell**

**Projet : MiniShell**  
**Auteur : Baudry Baptiste**

**1. Introduction**

Le projet **MiniShell** est une implémentation simplifiée d’un interpréteur de commandes UNIX, conçu pour exécuter des commandes de base, gérer les redirections, les pipes, et les commandes en arrière-plan. Ce shell prend également en charge des fonctionnalités avancées telles que la gestion des alias, l’historique des commandes, et l’exécution en mode batch.

L’objectif de ce document est de détailler la **logique de développement** du MiniShell en mettant en avant son architecture et les décisions techniques prises.

**2. Structure des fichiers**

Le projet est structuré en plusieurs fichiers pour garantir une meilleure **modularité** et **maintenabilité**.

| **Fichier** | **Rôle** |
| --- | --- |
| main.c | Point d’entrée du shell, initialise les composants |
| shell.c / shell.h | Gestion de la boucle principale et de l’entrée utilisateur |
| execute.c / execute.h | Exécution des commandes et gestion des processus |
| builtins.c / builtins.h | Implémentation des commandes internes (cd, exit, echo...) |
| alias.c / alias.h | Gestion des alias utilisateur |
| history.c / history.h | Sauvegarde et exécution de l’historique des commandes |
| env.c / env.h | Gestion des variables d’environnement (export, $VAR) |
| typedef.h | Définitions des structures de données utilisées |

**3. Fonctionnement et Logique du Shell**

**3.1. Lecture et Interprétation des Commandes**

Le shell suit une boucle d’exécution continue (shell\_loop()) :

1. Affiche un prompt (my\_shell> ).
2. Attend une entrée utilisateur (fgets()).
3. Stocke la commande dans l’historique (history.c).
4. Vérifie si la commande est un **builtin** (builtins.c).
5. Exécute la commande via execute\_logical\_commands().

**3.2. Exécution des Commandes**

Les commandes sont exécutées via execute\_single\_command() dans execute.c :

* Vérifie si la commande est un **builtin**.
* Utilise fork() pour créer un processus enfant.
* Le processus enfant exécute execvp() avec les arguments fournis.
* Le processus parent attend la fin de l’exécution (waitpid()), sauf si la commande est en arrière-plan (&).

**3.3. Gestion des Opérateurs Logiques (&&, ||)**

La fonction execute\_logical\_commands() analyse la chaîne de commande et exécute les sous-commandes dans le bon ordre :

* && : Exécute la seconde commande **uniquement si la première réussit** (return 0).
* || : Exécute la seconde commande **uniquement si la première échoue** (return != 0).

**3.4. Redirections (>, <, >>, <<)**

**Redirection de sortie (>, >>)**

Le shell ouvre un fichier et redirige STDOUT vers celui-ci

* > : Écrase le fichier existant.
* >> : Ajoute à la fin du fichier.

**Redirection d’entrée (<)**

Redirige l’entrée standard (STDIN) depuis un fichier

**Here-Document (<<)**

Lit des lignes jusqu’à un **délimiteur** et les stocke dans un fichier temporaire.

**3.5. Gestion des Pipes (|)**

Le shell utilise pipe() pour connecter plusieurs commandes :

* dup2(pipe\_fd[1], STDOUT\_FILENO) redirige la sortie du premier processus.
* dup2(pipe\_fd[0], STDIN\_FILENO) redirige l’entrée du second processus.

**3.6. Historique des Commandes**

Le fichier history.c stocke les commandes récentes :

* Écrit chaque commande dans .minishell\_history.
* Permet d’exécuter une commande précédente avec !N.

**3.7. Gestion des Alias**

Les alias permettent de remplacer une commande par une autre (alias ll="ls -al").

* Stockés dans une structure Alias.
* Remplacés avant exécution avec resolve\_alias().

**4. Mode Batch (-c "commande")**

Si ./my\_shell -c "ls -l" est utilisé :

* La commande est **exécutée immédiatement** sans entrer dans la boucle interactive.

**5. Conclusion**

Le projet **MiniShell** implémente une version simplifiée d’un shell UNIX en prenant en charge :

* **L’exécution des commandes simples et complexes**
* **Les opérateurs logiques (&&, ||) et les redirections**
* **Les pipes (|) et l’exécution en arrière-plan (&)**
* **L’historique et la gestion des alias**
* **Un mode batch (-c)**