Descriptif de la structure du code développé

I. Vue d'ensemble du projet:

Ce projet vise à implémenter les différentes étapes de la phase de réimplantation d'un éditeur de liens pour l'architecture ARM. Il est divisé en deux phases principales : l'analyse du fichier ELF (Phase 1) et la modification/création du fichier exécutable (Phase 2). Un ensemble complet de tests unitaires est fourni pour valider chaque étape.

II. Structure du code:

Le projet est organisé en trois répertoires :

- **include/:** Contient les fichiers d'en-tête (.h).
- **src/:** Contient les fichiers sources (.c).
- **test/:** Contient les scripts de test (.sh).

A. Répertoire include/ (Fichiers d'en-tête):

Chaque fichier d'en-tête définit les prototypes des fonctions et les structures de données pour une tâche spécifique :

- affichage_entete.h: Déclarations pour afficher l'en-tête ELF (Tâche 1).
- affichage_table_sections.h: Déclarations pour afficher la table des sections (Tâche 2).
- affichage_contenu_section.h: Déclarations pour afficher le contenu d'une section (Tâche 3).
- affichage_table_symboles.h: Déclarations pour afficher la table des symboles (Tâche 4).
- affichage_table_reimplantation.h: Déclarations pour afficher les tables de relocalisation (Tâche 5).
- renumeration_des_sections.h: Déclarations pour la renumérotation des sections (Tâche 6).
- correction_des_symboles.h: Déclarations pour la correction des symboles (Tâche 7).
- application_des_reimplantation.h: Déclarations pour l'application des reimplémentations R_ARM_ABS* (Tâche 8).
- call_jump.h: Déclarations pour l'application des reimplémentations R_ARM_JUMP24 et R_ARM_CALL (Tâche 9).

- modification_elf.h: Déclarations pour la fonction modification_elf (Tâches 6 à 9).
- executable.h: Déclarations pour la création du fichier exécutable (Tâche 11).

B. Répertoire src/ (Fichiers sources):

Le code source est organisé en modules fonctionnels :

- **Phase 1 (Tâches 1 à 5):** Analyse du fichier ELF.
    - program.c: Programme principal. Gère les options de ligne de commande et les appels aux fonctions d'affichage selon les arguments suivants :
        - -h: Affichage de l'en-tête ELF.
        - -S: Affichage de la table des sections.
        - -s: Affichage de la table des symboles.
        - -r: Affichage des tables de relocalisation.
        - -x :Affichage du contenu d'une section spécifique.
    - affichage_entete.c, affichage_table_sections.c, affichage_contenu_section.c, affichage_table_symboles.c, affichage_table_reimplantation.c: Implémentations des fonctions d'affichage.
- **Phase 2 (Tâches 6 à 11):** Modification du fichier ELF et création de l'exécutable.
    - program2.c: Programme principal. Appelle la fonction modification_elf.
    - renumeration_des_sections.c, correction_des_symboles.c, application_des_reimplantation.c, call_jump.c: Implémentations des fonctions de modification du fichier ELF.
    - modification_elf.c: Fonction principale qui orchestre les modifications du fichier ELF.
    - executable.c: Implémentation de la création du fichier exécutable.
- **Tâche 10:** Interaction avec le simulateur.
    - ARM_elf.c: Code pour l'interaction avec le simulateur ARM.

C. Répertoire test/ (Tests):

Ce répertoire contient les scripts de test, organisés par tâche :

- test_header/test1_header.sh: Test de l'affichage de l'en-tête (Tâche 1). Argument : <chemin_fichier_elf>.

- test_index_table/test_index_table.sh: Test de l'affichage de la table des sections (Tâche 2). Argument : <chemin_fichier_elf>.
- test_print_section_table/test_print_section.sh: Test de l'affichage du contenu d'une section (Tâche 3). Arguments : <chemin_fichier_elf> <numéro_section>.
- test_affichage_table_de_symbol/test_affichage_table_symbole.sh: Test de l'affichage de la table des symboles (Tâche 4).
  Argument : <chemin_fichier_elf>.
- test_realocation_table/test_realocation_table.sh: Test de l'affichage des tables de relocalisation (Tâche 5). Argument : <chemin_fichier_elf>.
- test_renumeration_des_sections/test_renumeration_des_sections.sh: Test de la renumérotation des sections (Tâche 6). Argument : <chemin_fichier_elf>.
- test_correction_des_symboles/test_correction_des_symboles.sh: Test de la correction des symboles (Tâche 7). Argument : <chemin_fichier_elf>.
- Reimplantations_de_type_R_ARM_ABS/test.sh: Test de l'application des reimplémentations R_ARM_ABS* (Tâche 8). Argument : <chemin_fichier_elf>.
- Reimplantations_de_type_R_ARM_JUMP24_et_R_ARM_CALL/test.sh: Test de l'application des reimplémentations R_ARM_JUMP24 et R_ARM_CALL (Tâche 9). Argument : <chemin_fichier_elf>.
- automated_testing.sh: Lance tous les tests. Argument optionnel : <chemin_fichier_elf> (si spécifié, teste uniquement avec ce fichier).(Pour faire les tests il faut exécuter la commande ./test/automated_testing.sh avec le numero du test qu'on souhaite effectuer) .

IV. Script program_run.sh:

Ce script lance le simulateur ARM (./arm_simulator ...) et exécute ensuite ./ARM_elf pour charger et exécuter le fichier ELF spécifié en argument ($output).

Cette description est organisée de manière claire et précise, facilitant la compréhension de l'architecture et du fonctionnement du projet. Elle est prête pour une intégration directe dans un document.