STAGE - DÉVELOPPEMENT LOGICIEL ET OPTIMISATION ALGORITHMIQUE

A.I.Mergence

15 Juin 2020

1 Introduction

Dans le cadre du recrutement des futures stagiaires A.I.Mergence organise des tests techniques. L'objectif de ces tests n'est pas de vous faire peur mais d'instaurer un échange avec un potentiel collaborateur. Outre les compétences techniques, ces tests ont pour but d'évaluer vos méthodes de résolution de problèmes. Le but ici n'est pas forcément d'arriver à tout finir mais de mettre en oeuvre les bonnes stratégies de raisonnement utiles à A.I.Mergence.

1.1 Fonctionnement du test

Chacun des exercices du test est indépendant. Si vous vous retrouvez bloqué sur un exercice, n'hésitez pas à passer au suivant. Chacun des exercices se situe dans un dossier séparé. Ainsi vous trouverez les dossiers suivants:

- **1_connaissance_systeme_linux:** Contient l'exercice 1 sur la compréhension des commandes unix.
- **2_algorithmes_de_tri:** Contient l'exercice 2 concernant le tri de données.
- **3_application_reel:** Contient l'exercice 3 concernant le tri de données sur un problème réel.

2 Connaissance du système linux

L'objectif de cet exercice est d'évaluer vos connaissances dans l'utilisation des commandes d'un système UNIX.

2.1 Objectif:

Pour des raisons obscures des données importantes se sont retrouvées éparpillées dans divers fichiers textes. L'objectif de cet exercice est de créer un script bash ou python permettant de concaténer de manière récursive tous les fichiers textes

(et exclusivement les fichiers textes) présents dans un dossier en grand fichier texte final. Le dossier 'data' contient les données éparpillées.

2.2 Indice:

Une solution simple et efficace est l'utilisation d'une boucle for.

2.3 Résultats attendus:

2.3.1 Si vous avez choisi bash

./concatenation.sh → Toutes les données ont étés regroupées dans le fichier data.txt

2.3.2 Si vous avez choisi python

python3 concatenation.py → Toutes les données ont étés regroupées dans le fichier data.txt

2.4 En cas de données corrompues:

Lancer python3 init_data.py

3 Création d'un algorithme de tri

3.1 Objectif:

L'objectif de cet exercice est de créer un algorithme de tri de données pouvant être exécuté sur plateforme embarquée. Le tri doit s'effectuer par ordre croissant sur des données alphanumériques en un minimum de temps.

3.2 Exemple:

"iJvhKbF4peEtzexnCfnDBAg810OGwI" → "0148ABCDEFGIJKObeefghinnptvwxz"

3.3 Indice:

L'utilisation d'un algorithme de tri rapide est fortement conseillé. Petit lien wikipedia https://fr.wikipedia.org/wiki/Tri_rapide

3.4 Résultat attendu:

./sort data → Les données ont été triées

4 Application à un problème réel

L'objectif de cet exercice est d'appliquer l'algorithme de tri créé dans l'exercice 2 à un problème réel.

4.1 Objectif:

Dans le dossier de cet exercice vous trouverez le fichier "data.txt". Ce fichier contient n blocs de 1024 octets. Le but de cet exercice est de lire ces blocs, de les trier et de les sauvegarder dans un fichier séparé. Comme l'exercice 2, cet exercice doit pouvoir s'exécuter sur plateforme embarquée en un minimum de temps.

4.2 Si vous n'avez pas fini l'exercice 2:

Un algorithme de tri est déjà implémenté. Vous le trouverez dans include/Sort.h et dans src/Sort.cpp

4.3 Indice:

L'utilisation de threads est fortement conseillée.

4.4 Résultat attendu:

./sort nom_du_fichier → Le fichier a été trié