Titre : Architecture du serveur de matrice de couleurs

Introduction

Ce document décrit l'architecture d'un serveur de matrice de couleurs en C. Le serveur maintient une matrice 2D de couleurs simples et permet aux clients de se connecter et d'effectuer des opérations telles que la définition de pixels, l'obtention de la taille de la matrice et la récupération de la version du protocole.

Fonctionnalités principales

Connexion des clients au serveur

Gestion des commandes des clients pour modifier la matrice de couleurs

Envoi de la matrice de couleurs mise à jour aux clients

Limitation du taux de modification des pixels pour chaque client

Gestion des commandes pour obtenir des informations sur la matrice et le serveur

Structures de données

Structure client : stocke les informations sur chaque client connecté, y compris le socket, l'adresse, le nombre de commandes setPixel envoyées et le moment de la dernière réinitialisation de la limite de commandes.

Structure color: stocke les informations sur une couleur, y compris les composantes rouge, verte et bleue.

Fonctions principales

to_base64(int value): convertit une valeur en caractère base64.

from_base64(char base64_char): convertit un caractère base64 en valeur.

remove_client(int sd, struct client **clients): supprime un client de la liste chaînée.

create_matrix(int width, int height): crée une matrice de couleurs de taille spécifiée.

matrix_to_string(Color **matrix, int width, int height): convertit la matrice de couleurs en une chaîne de caractères.

set_pixel(Color **matrix, int x, int y, int r, int g, int b, int width, int height): modifie la couleur d'un pixel spécifié dans la matrice.

main(int argc, char *argv[]): fonction principale du programme.

Gestion des connexions

Le serveur utilise les fonctions socket(), bind(), listen(), et accept() pour créer un socket de serveur, lier l'adresse et le port, écouter les connexions entrantes et accepter les nouvelles connexions. Les

clients sont ajoutés à une liste chaînée de clients et leurs sockets sont ajoutés à un ensemble de descripteurs de fichiers pour la sélection.

Traitement des commandes

Le serveur utilise la fonction select() pour attendre et détecter les activités sur les sockets. Pour chaque socket actif, il lit les données entrantes et traite les commandes suivantes:

/getWaitTime: envoie le temps restant avant la réinitialisation de la limite de commandes setPixel.

/getVersion: envoie la version du protocole utilisé par le serveur.

/getSize: envoie la taille de la matrice de couleurs.

/getLimits: envoie le nombre de commandes setPixel restantes pour le client.

/setPixel: modifie la couleur d'un pixel spécifié dans la matrice, si le client n'a pas atteint la limite de commandes.

Envoi des mises à jour

Après avoir traité les commandes, le serveur convertit la matrice de couleurs en chaînede caractères et l'envoie à tous les clients connectés pour mettre à jour leur affichage de la matrice.

Limitation du taux de modification des pixels

Pour prévenir les abus et les surcharges du serveur, une limite est mise en place sur le nombre de commandes setPixel qu'un client peut envoyer dans un intervalle de temps donné. Cette limite est vérifiée à chaque fois qu'un client tente d'envoyer une commande setPixel. Si la limite est atteinte, le serveur refuse d'appliquer les modifications demandées et envoie un message d'erreur au client. La limite est réinitialisée périodiquement pour chaque client.

Gestion des erreurs et déconnexions

Le serveur gère les erreurs de communication avec les clients en vérifiant le statut de retour des fonctions recv() et send(). Si une erreur se produit ou si un client se déconnecte, le serveur supprime le client de la liste chaînée et ferme le socket correspondant.

Conclusion

L'architecture du serveur de matrice de couleurs décrite dans ce document permet de créer un système simple et efficace pour gérer une matrice de couleurs partagée entre plusieurs clients. Les fonctionnalités principales incluent la connexion des clients, la gestion des commandes pour modifier la matrice, l'envoi de la matrice mise à jour aux clients et la limitation du taux de modification des pixels. Cette architecture peut être étendue et améliorée pour prendre en charge des fonctionnalités supplémentaires et optimiser les performances.