Projet de l'atelier d'ingénierie (OS-Git) pour la filière SUD CLOUD

Réalisation d'un Simulateur de Réseau Token Ring

Réalisé par :

EL-HLOU Salma

AHORRAR Omar

ASSKOUR Mohamed

Voici le lien du projet de l'atelier sur Git :

Git-Link.

1. INTRODUCTION :

Le Token Ring est une topologie de réseau associée à un protocole de la couche liaison du modèle OSI. Ce protocole fonctionne en circulant un jeton (token) dans une direction unique autour d'un anneau. Chaque station peut transmettre des données uniquement lorsqu'elle possède le jeton. Le projet a consisté à implémenter un simulateur de ce réseau en utilisant des pipes pour la communication inter-processus.

Ce rapport présente l'approche adoptée, les défis rencontrés et une analyse des résultats obtenus.

2. Objectifs du projet:

0.1 Implémenter un simulateur de réseau Token Ring :

- Permettre la circulation d'un jeton dans un réseau circulaire.
- Simuler l'émission de messages par les nœuds lorsqu'ils possèdent le jeton.

0.2 Respecter les spécifications suivantes :

- Le nombre de nœuds doit être paramétrable.

- Chaque nœud doit attendre le jeton avant d'envoyer des messages ou de le transmettre.

- Utiliser des pipes comme moyen de communication.

0.3 Documenter l'approche technique et les résultats.

3. Conception et implémentation :

0.4 Architecture du simulateur :

Nœuds : Chaque processus représente un nœud du réseau.
Pipes : Utilisés pour la communication unidirectionnelle entre les nœuds.
Parent : Le processus parent initialise le réseau et injecte le premier jeton.

0.5 Fonctionnement des nœuds

Chaque nœud effectue les opérations suivantes :
Attente du jeton : Lecture du jeton via son pipe d'entrée.
Transmission de messages : Décide de transmettre un message ou de passer directement le jeton.
Passage du jeton : Écrit le jeton dans le pipe de sortie vers le nœud suivant.

0.6 Gestion des pipes :

- Un tableau de pipes connecte les nœuds entre eux dans une structure circulaire.
- Le dernier nœud est relié au premier pour former un anneau.

0.7 Analyse du code fourni :

Le code utilise un tableau de pipes pour connecter les processus entre eux. Chaque processus enfant représente un nœud et exécute les étapes suivantes :

0.7.1 1. Initialisation des pipes :

Chaque nœud ferme les extrémités inutiles des pipes pour garantir une communication isolée.

0.7.2 2. Traitement du jeton :

Le nœud lit le jeton depuis son pipe d'entrée, affiche un message, incrémente la valeur du jeton et le transmet au nœud suivant.

0.7.3 3. Condition de terminaison :

Le jeton revient à 0, ce qui provoque la sortie de la boucle et termine le processus enfant.

0.7.4 4. Processus parent :

Le processus parent initialise le jeton et attend la fin de tous les processus enfants.

0.8 Concernant les commandes Git :

1. Sous Linux, nous avons créé le répertoire principal du projet : TokenRingTP
2. Nous avons initialisé ce répertoire en tant que dépôt Git avec la commande : git init.
3. Nous avons créé notre fichier source principal : Token.c
4. Nous l'avons ajouté à la phase indexée avec la commande : git add .
5. Nous l'avons enregistré avec la commande : git commit -m "Token.c"
6. Nous avons poussé le dépôt sur GitHub avec la commande : git push -u origin main

4. Code TOKEN.C :

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int N ;

int main() {
    int pipes[N][2];
    int pid;
    int token = 1;

    for (int i = 0; i < N; i++) {
        if (pipe(pipes[i]) == -1) {
            return -1;
        }
    }

    for (int i = 0; i < N; i++) {
        pid = fork();
        if (pid == -1) {
            return -1;
        }

        if (pid == 0) {
            int buff;

            for (int j = 0; j < N; j++) {
                if (j != i) close(pipes[j][0]);
                if (j != (i + N - 1) % N) close(pipes[j][1]);
            }
            while (1) {
                read(pipes[i][0], &buff, sizeof(int));
                printf("Noeud␣%d␣a␣re u␣le␣jeton␣%d\n", i + 1, buff);

                sleep(1);

                buff = (buff + 1) % N;
                write(pipes[(i + 1) % N][1], &buff, sizeof(int));

                if (buff == 0) break;
            }

            exit(0);
```

```
        }
    }

    for (int i = 0; i < N; i++) {
        close(pipes[i][0]);
        if (i != 0) close(pipes[i][1]);
    }

    write(pipes[0][1], &token, sizeof(int));

    for (int i = 0; i < N; i++) {
        wait(NULL);
    }

    return 0;
}
```

5. Résultats et validation (Réseau avec six nœuds)

Exemple de sortie :

- Nœud 1 : Reçu le jeton 1
- Nœud 1 : Transmission du jeton 2
- Nœud 2 : Reçu le jeton 2
- Nœud 2 : Transmission du jeton 3
- Nœud 3 : Reçu le jeton 3
- Nœud 3 : Pas de message à envoyer
- Nœud 4 : Reçu le jeton 4
- Nœud 4 : Transmission du jeton 5
- Nœud 5 : Reçu le jeton 5
- Nœud 5 : Transmission du jeton 6
- Nœud 6 : Reçu le jeton 6
- Nœud 6 : Transmission du jeton 0

6. CONCLUSION :

Le simulateur de réseau Token Ring réalisé répond aux objectifs fixés. Il reproduit fidèlement le fonctionnement d'un réseau à jeton en utilisant des pipes pour la communication entre les processus.
Ce projet a permis de renforcer nos compétences en programmation système, en particulier dans la gestion des processus et la communication inter-processus. Il ouvre la voie à une meilleure compréhension des mécanismes internes des réseaux et des protocoles de communication.