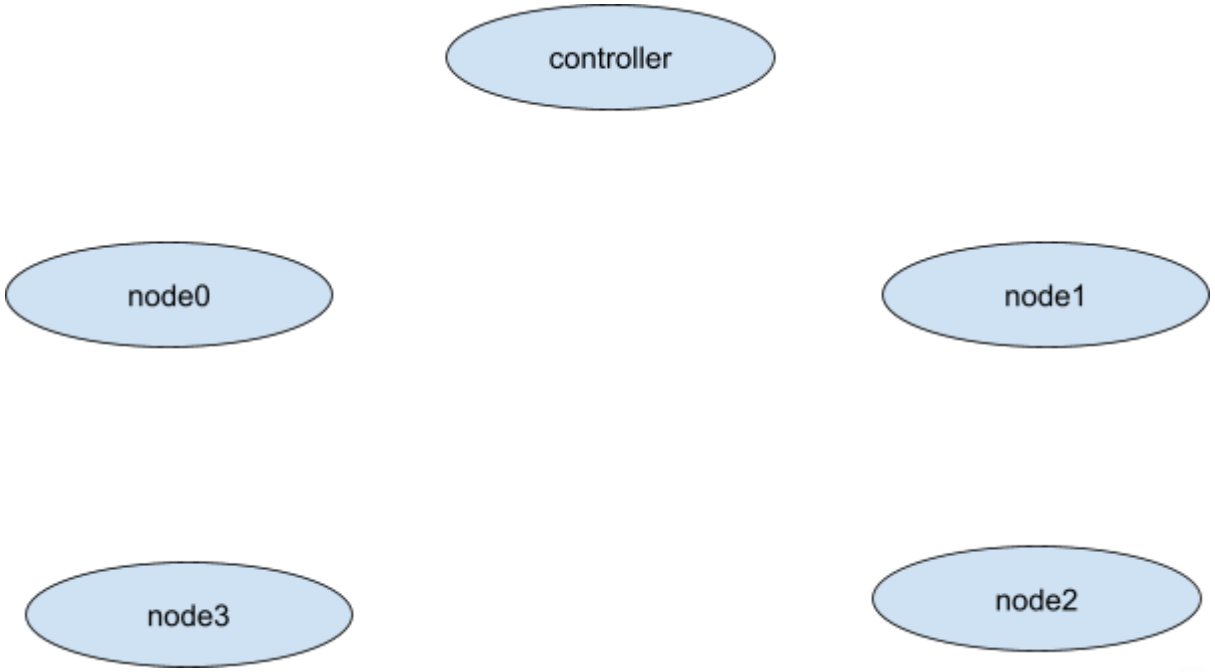
Rapport de projet système

1.Conception

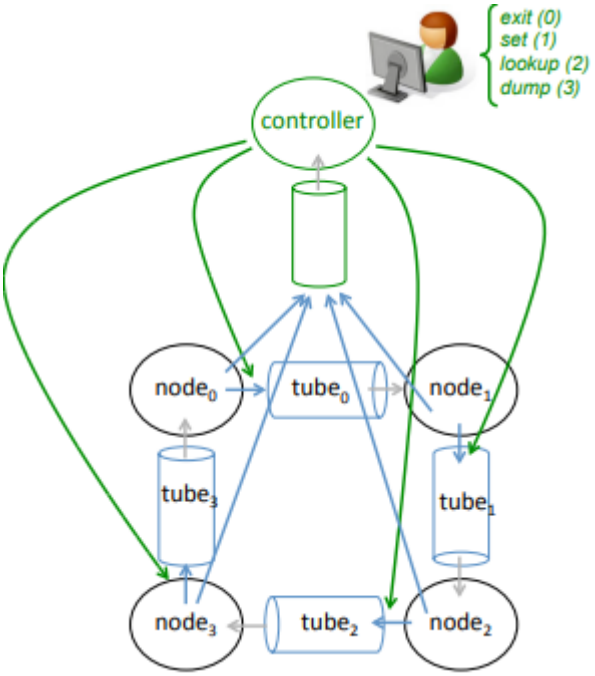
1.1 La création des nodes

D'abord on va créer n nodes par programme .Chaque node est un processus fils du processus principal —- contrôler .Dans le node fils ,il y a une liste chaînée qui enregistre un mot avec sa clé .Et le contrôler ,c'est utilisateur .On peut exécuter une commande par le contrôler ,on décide le nombre de node fils par le controller aussi .



1.2 La création des pipes

Pour réaliser la communication entre des nodes ,on utilise le pipe .On connecte des nodes en structure d'anneau ,c'est à dire on fait un cycle entre tous les nodes ,il y a toujours un pipe entre node(i) et node(i-1) ,node(i) et node (i+1) ,chaque node lit les infos venant du node précédent et envoie les infos à la node suivante si ce n'est pas lui qui doit les traiter .Bien sur, le contrôleur peut recevoir les infos par un pipe(pipe(n)) et écrire la commande à chaque pipe .Donc ici on a besoin n+1 pipes.



1.3 Exit

Pour réaliser la commande exit ,on envoie la commande à chaque node en même temps ,et on tue tous les processus .

1.4 Set

Pour enregistrer un mot avec une nouvelle clé ,on envoie les infos(commande ,clé, mot) à node(0)par le tube (pipe n-1),puis si le clé est dans cette node ,on exécute la méthode store() et renvoie le résultat de réussir au contrôleur; si le clé n'existe pas dans la liste chaînée ,le clé est déjà existé, on envoie un résultat échoué ,sinon ,la node(0) envoie les infos à node(1) ,puis node(1) fait la même chose jusqu'à node(n-1) .

1.5 Lookup

C'est pareil que Set .D'abord on envoie les infos à node(0) ,si la clé doit être traité par cette node et qu'elle existe ,on renvoie le résultat réussir avec la valeur via le tube du contrôleur ,si elle n'existe pas ,on renvoie un résultat échoué ,sinon la node envoie les infos à la node suivante .

1.6 Dump

C'est pareil que exit .On envoie la commande à chacun en même temps et affiche toutes les tables .Mais il doit afficher par ordre ,donc la node suivante doit attendre la fin d'exécution de la node précédente .

2.Fonctions

2.1 Commande

Définition de commande et méthodes :

```
typedef struct Connamde{
    int command;
    int key;
    char data[128];
} Commande;

void init_commande(Commande* commamde);
```

```
void new_commande(Commande* commamde);
```
int command :la commande (de 0 à 3 ,exit ,set ,lookup ,dump)

int key :la clé

char data[128] :Pour set ,premièrement ,ici c'est le mot qu'on va enregistrer ,puis on envoie tous la commande au processus fils .Après la réussite de set ,ici c'est le résultat "store ok" puis on affiche ce résultat dans le controller .
Pour lookup ,premièrement c'est nul .Après lookup ,c'est le résultat(clé existe) ou nul(clé n'existe pas) .
C'est-à-dire ,le data c'est le mot(avant exécution de la méthode du tableau) et le résultat(après exécution) .

init_commande() :initialise la commande
new_commande() :renouvelle la commande

2.2 Fonction

Ici la définition :
```
void close_pipes(int** pipes, int indice, int n);
void list(Commande* commande, int reading_pipe[], int writing_pipe[],
int controller_pipe[], int n, int indice);
void controller(Commande* commande, int** pipes, int n);
```

close_pipes() :Ferme les pipes inutiles .
Par exemple ,pour node(0) ,on n' ouvre que le dernier tube pipe(n-1) pour lire l'info voire la requête qui vient de pipe(n-1) ,et on écrit qu' à pipe(0) pour transmettre au node suivante ou au tube du controller pipe(n) pour transmettre au controller .

list() :Créer une liste chainée dans chaque node ,puis faire l'exécution de la commande .
Pour réaliser cette capacité ,d'abord on initialise un tableau ,puis lit le 1er commande et fait l'exécution ,ensuite c'est le 2eme ,3eme … Donc on utilise la boucle do…while ici .On fait la lecture et l'exécution de chaque commande jusqu'à la commande exit ,puis on tue les processus fils ici .

controller :Créer des commandes et les envoyer .Pour créer des commandes ,on utilise la boucle do…while encore .On crée une commande d'abord , si 0 ou 3 ,on l'envoie à chacun ,sinon la node 0 ,puis reçoit le résultat et l'affiche .Si la commande est exit ,on sort de la boucle .

3.La solution de synchronisation de dump

Pour la commande dump ,si on ne fait pas une solution de synchronisation ,l’affichage ne sera pas dans le bon ordre ,donc on a besoin de la solution de synchronisation pour éviter ce problème .
Donc on donne une solution ici :pour faire l'exécution de dump ,il a besoin de 2 infos ,l'une est la commande vient du controller et l’autre est le résultat réussi vient de node précédente. D’abord le contrôleur envoie la commande dump a chacun ,puis tous les nodes sauf la node(0) attend ,comme il n’y a pas de résultat réussi vient de la node précédente ,et la node(0) fait l'exécution de la commande comme elle est la première node .Quand la node(0) fini l’affichage ,elle envoie le résultat à node suivante et contrôler ,c’est node(1) ,puis node(1) fait la même chose ,jusqu'à la node(n) .A la fin ,le controller compte la nombre de résultat ,si égal à n,c'est-à-dire la commande dump est fini ,on peut recommence l’interaction avec l’utilisateur .