Université de Reims
Champagne-Ardenne
U.F.R. de Sciences
Exactes et Naturelles

Licence Info
Info(rmatique)0403
2023/2024

Travaux Pratiques 4 et 5

à rendre au plus tard le Mercredi 3 Avril 2024
(Compte Rendu + Code Source)

On souhaite mettre en place un système de processus structurés en hypercube. Les hypercubes sont une famille de graphes. Dans un hypercube Q_n (n étant la dimension de l'hypercube qui sera passé en paramètre à votre programme), chaque sommet du graphe porte une étiquette de longueur n sur un alphabet $A = \{0, 1\}$, et deux sommets sont adjacents (ie il existe une arête entre eux) si leurs étiquettes ne diffèrent que d'un symbole (par exemple 0010 et 1010 pour un hypercube de dimension 4). Un hypercube de dimension 2 est un carré ($n = 2$) et un hypercube de dimension 3 est un cube ($n = 3$).

Dans les années 1980, des ordinateurs furent réalisés avec plusieurs processeurs connectés selon un hypercube : chaque processeur traite une partie des données et ainsi les données sont traitées par plusieurs processeurs à la fois, ce qui constitue un calcul parallèle. L'hypercube est couramment introduit pour illustrer des algorithmes parallèles, et de nombreuses variantes ont été proposées, soit pour des cas pratiques liés à la construction de machines parallèles, soit comme objets théoriques.

La construction de l'hypercube devra être l'étape préalable de votre projet. Un hypercube de dimension n comporte 2^n sommets et $2^{n-1} \times n$ arêtes (une arête étant la possibilité pour les deux sommets adjacents à l'arête de communiquer entre eux, elle se "transformera" en 2 tubes). La page wikipedia sur l'hypercube vous détaillera précisement sa structure : https ://fr.wikipedia.org/wiki/Hypercube_(graphe).

Vous devez réaliser un parcours sur un hypercube de dimension n (passé en paramètre au démarrage du programme). Si deux sommets de l'hypercube sont adjacents, vous devez permettre à ces deux processus (représentés comme sommets du graphe) de communiquer en utilisant des tubes (deux précisémment : de l'un à l'autre et de l'autre à l'un). Pour identifier les processus, vous utilisez des étiquettes de tailles n (telles que décrites précédemment) : deux tubes existent donc que si un seul caractère diffère entre les étiquettes des deux processus concernés.

L'ensemble des processus participant à ce système communique grâce à des tubes et va faire circuler un message qu'on appellera par la suite le *jeton* (*i.e. token*). Chaque processus créera un fichier (ayant pour nom l'étiquette du processus) et stockera dans ce fichier les dates des visites successives du jeton.

Chaque processus devra donc attendre la réception du jeton. Pour réaliser cela, il faudra qu'un processus puisse lire plusieurs descripteurs de fichier *en même temps.* La fonction `int select(int nfds, fd_set * readfds, fd_set * writefds, fd_set * exceptfds, struct timeval *timeout);` permet de consulter pendant une période de temps égale à *timeout* trois ensembles de descripteurs de fichier ($readfds$, $writefds$ et $exeptfds$) et retourne le nombre de descripteurs de fichiers disponibles. Les ensembles de descripteurs de fichiers $readfds$, $writefds$ et $exeptfds$ sont modifiés et ne conservent que les descripteurs de fichiers dont le statut a été modifié (Dans le cadre de ce projet, vous aurez à utiliser uniquement $readfds$, les autres $writefds$ et $exceptfds$ seront totalement inutiles). Si la valeur $NULL$ est fournie en guise de *timeout* c'est un test immédiat (Dans le cadre de ce que vous avez à faire, il faudra faire une boucle d'attente sur la fonction *select*).

Le type fd_set est un ensemble de dscripteurs de fichiers et se manipule avec els fonctions suivantes :

— `void FD_ZERO(fd_set *set);` Initialise un ensemble de descripteur vide.
— `void FD_SET(int fd, fd_set *set);` Ajoute le descripteur fd à l'ensemble *set*.
— `void FD_CLR(int fd, fd_set *set);` Retire le descripteur fd à l'ensemble *set*.
— `int FD_ISSET(int fd, fd_set *set);` Teste si fd appartient à l'ensemble *set*.

Le parcours de l'hypercube doit être réalisé de manière infinie et aléatoire (au démarrage, le processus $00 \ldots 0$ enverra un message vers l'un de ses processus adjacents choisi au hasard. Ce processus redirigera lui aussi ce message de la même manière.

Pour chaque processus, et pour chaque visite, vous enregistrerez le nombre de microsecondes écoulées entre cette visite et la précédente (sur le même processus) dans le fichier dédié à ce processus.

Le programme doit donc s'exécuter avec la commande : \$./*prog* n où n est le paramètre du programme (ie la dimension de l'hypercube).

Le père (qu'on peut imaginer être le processus d'étiquete $00\ldots0$, à vous de voir ...) peut, à la demande de l'utilisateur, suspendre l'ensemble des processus (en utilisant un signal $SIGSTOP$ vers chacun de ceux-ci). L'ensemble des processus est donc suspendu jusqu'à ce que l'utilisateur demande au processus père de reprendre l'exécution (signal $SIGCONT$). Pour communiquer avec le processus père, l'utilisateur lui enverra des signaux $SIGUSR1$ (le premier suspend les processus fils, le deuxième les remet en cours d'exécution, et ainsi de suite...).

La terminaison de l'ensemble de processus devra elle aussi être piloter par des signaux (tous les processus attendent de recevoir le message jeton). Il suffit alors d'envoyer un signal donné au processus père qui relaiera à tous les processus fils.

Vous devez réaliser ce programme. La mise en place de la structure de communication doit être effective avant de démarrer le parcours du jeton *Token Hypercube Traversal*.

Vous devrez fournir le code et un compte-rendu (env. 8 pages) par binôme.

La grille d'évaluation prendra en compte (à la fois dans le code et dans le rapport) :
— la mise en place de la structure de communication.
— le parcours du jeton.
— la gestion des signaux.
— l'enregistrement des fichiers.
Par ailleurs une attention sera portée à :
— Si votre code est lisible.
— Si votre programme compile correctement.
— Si votre programme s'execute sans erreur de segmentation.