UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

06/04/2022

TOKEN RING

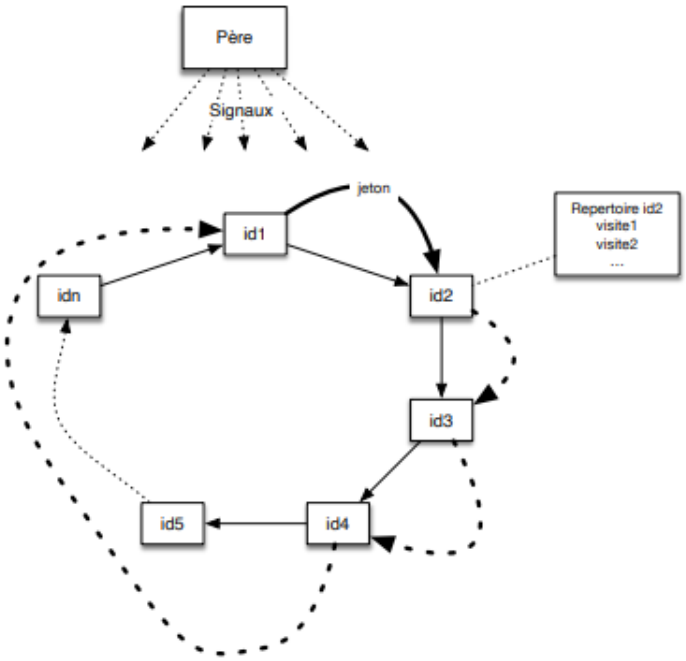
PROJET INFO403

SAMI DRIOUCHE & WALID AIT-ERRAMI
S4F3B

Table des matières

Introduction

Le TP avait pour but de créer et de mettre en place un système Token Ring entre différents processus au travers de l'architecture donnée ci dessous.



Pour cela on a utilisé et manipulé plusieurs mécanismes qu'on appris tout le long de notre semestre comme les tubes , les processus et même les signaux .
On va donc approfondir tout d'abord sur la création de l'anneau jusqu'à l'utilisation du jeton .

Puis on continuera notre analyse sur les sauts de ce jeton entre chaque processus et comment on a fais pour avoir une trace du temps de passage de ce jeton . Ensuite nous allons voire comment nous avons sauvegarder ses temps de passage grâce aux fichiers qui sont dans différents répertoires .

Enfin nous allons voire la partie qui est origine d'arrêt et de continuation mais surtout de communication utilisateur et processus , nous parlions en effet des signaux .

L’anneau et le jeton

Notre anneau est composé de n processus fils qui sera créer grâce a notre seule et unique processus père grâce à l’aide de la méthode fork(). Entre ses processus nous aurons la présence de tubes qui serons la pour la communication de données entre chaque processus fils . La création de processus fils ne serons confirmer avec cette affichage qui nous montre le numéro du processus créer et son pid . Ici nous avons lancer le programme avec la création de 10 processus créé .

Pour la création des tubes nous avons créer donc n tubes et donc tubes de [n] [2] afin d’avoir les deux case pour l’émission et la réception de données.
Pour les créer nous avons fait une boucle for qui va jusqu’à n afin d’avoir un tube entre deux processus. La case réception sera remplacer par le int in et l’émission par out. N’oublions aussi que nous avons fermer au préalable les case des tubes que nous avons pas besoin au fur, et à mesure selon la valeur d’un booléen .
Pour le jeton , c’est son émission et sa réception qui sera important pour celui ci . En effet le jeton n’est nul guère un int qui est envoyer dans l’anneau par les tubes entre les processus .

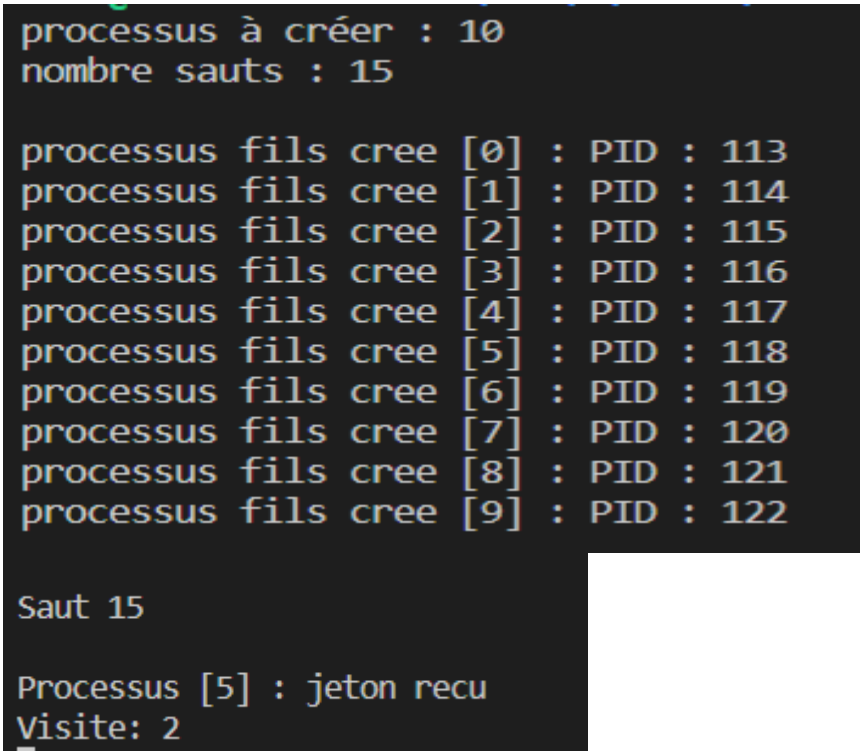
Pour cela nous avons mis le out au tube d’indice (x+1)%n , ainsi le x-ième processus enverra son message dans le tube d'indice (x+1)%n et lira dans le tube x, afin de créer un anneau.
Les fonctions read (in ….) et write (out…) seront les appels qui seront ceux qui envoient et recevront le jeton entre chaque processus , par les tubes . Le booléen bol n’est la seulement pour savoir si on n’est seulement à la fin du tour de l’anneau ou pas .
La réception et l’émission du jeton sera confirmer avec un affichage très détaillé .

```
Processus [0] :  envoie du jeton

Saut 1

Processus [1] : jeton recu
Visite: 1
Processus [1] :  envoie du jeton
```

Comme on peut le voire ci-dessus , voici un exemple de l'affichage que je vous ai parler . On peut voire ici que le processus 0 qui est donc le premier processus fils envoie le jeton , le saut (cf voire partie 2) est donc effectuer . Le jeton est donc réceptionné par le processus 1 qui lui même renvoie le jeton .

Les sauts et le temps de passage

Les sauts sont les nombres de passage du jeton à travers les tubes . Ce nombre m est le deuxième et le dernier argument qui est demander lors de l'exécution du programme. En effet, on vérifie après qu'il n'y a aucun autre argument après celui-ci. Ce saut sera incrémenté lorsque le jeton sera émis vers un autre processus . Avant chaque émission du jeton , on vérifie que le saut dépasse le nombre de saut m, si celui dépasse m alors le booléen prend la valeur 1 ce qui veut dire qu'on ne pourra plus envoyer le jeton à travers aucun tubes . Si ce m n'est pas dépassé alors les sauts de jeton continue..



Dans cette exemple ci-contre nous avons exécuter le programme avec 10 processus fils à créer et 15 sauts de jetons à réaliser .
On a bien 10 processus qui ont été bien créer et à la fin du programme 15 sauts on été réalisé .

Maintenant parlons du temps de passage , le temps de passage est le temps que fais le jeton de passer vers un processus ,faire le tour de l'anneau et revenir vers ce même processus . Pour calculer cela nous avons créer trois variables , une qui aura le temps exacte du temps de passage qui sera grâce a l'aide des deux autres variables qui auront pour but de sauvegarder deux laps de temps précis et faire donc la différence de ces deux checkpoint de temps pour avoir le temps exacte. Pour avoir les deux checkpoint de temps nous avons utiliser la méthode gettimeofday() afin de récupérer le temps exacte de ce moment précis dans le programme.

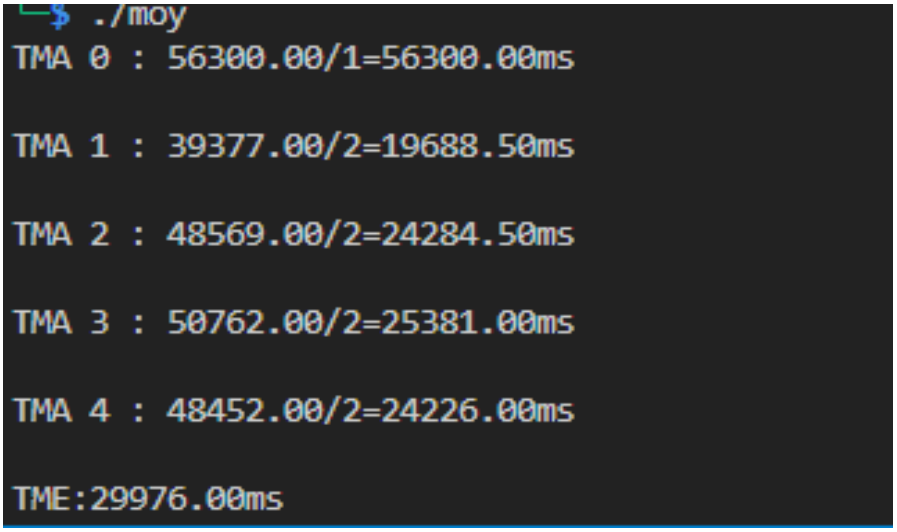
On verra dans la partie suivante que ces temps de passage seront sauvegardé dans des fichiers …

Les fichiers et répertoires

Les fichiers sont créer selon le nombre de visite d'un processus que réalise le programme , si le jeton passe 5 fois sur un processus par exemple , alors on aura 5 visites et donc 5 fichiers. Ces fichiers seront ranger dans un répertoire .

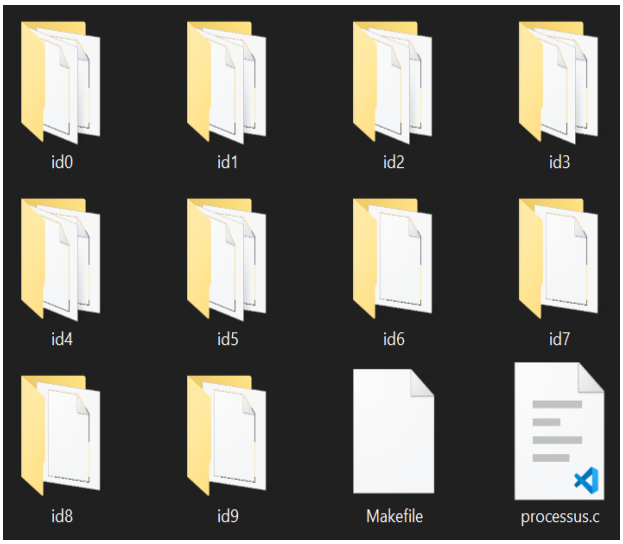
Chaque processus aura son propre répertoire et ses propres fichiers selon le nombre de ses visites . Le temps de passage sera inscrit dans le fichier après son ouverture . On vérifie encore que le nombre de saut qu'il reste avant de l'écrire dans le fichier. Ce nombre de visite sera incrémenté après chaque émission (en mème temps que incrémentation du saut) .

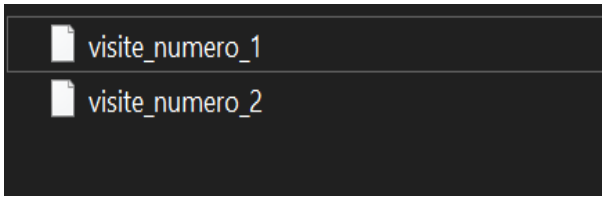
Ces fichiers seront inscrit dans un répertoire qui sera ouvert dans un autre programme afin de récupérer tout les temps de passage dans tout les fichiers de ce répertoire afin de calculer en outre la moyenne des temps de visite par processus et la moyenne générale sur l'ensemble des processus.



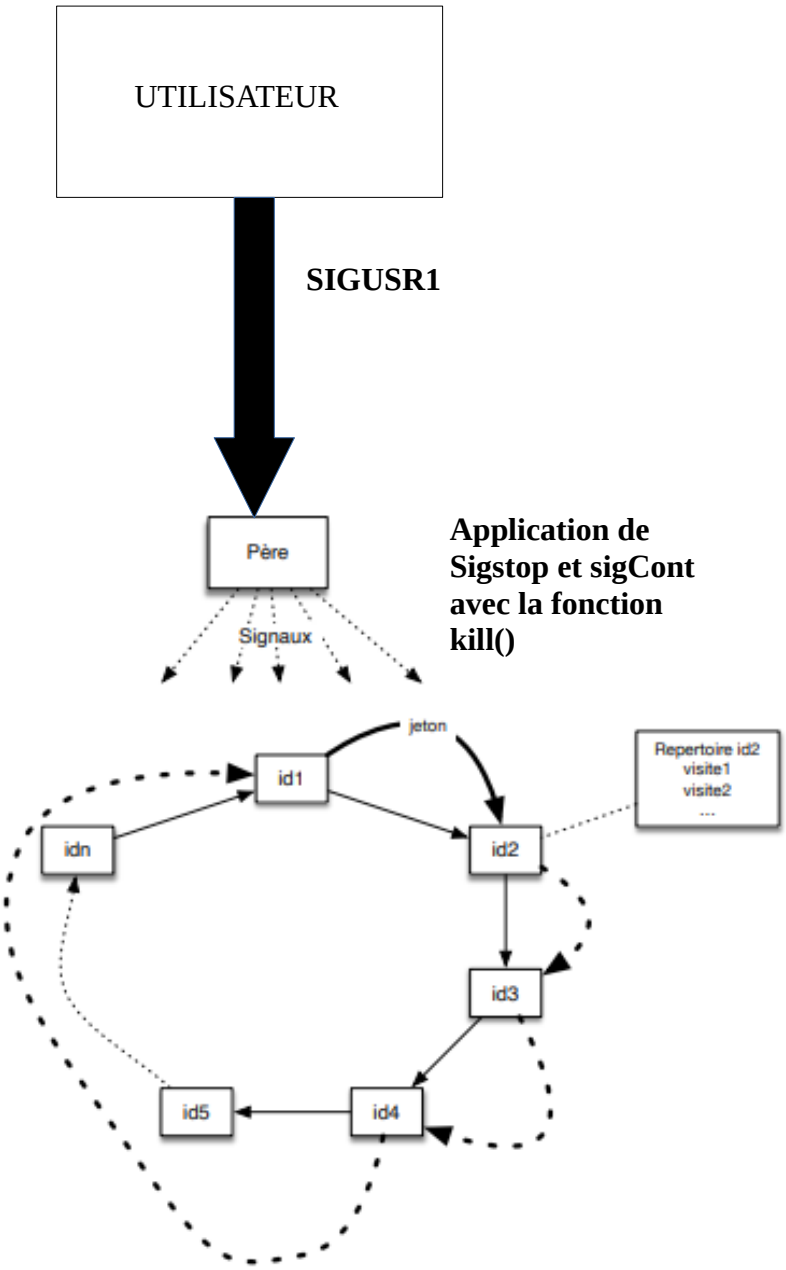
Les ouvertures de répertoire et de leur manipulation se feront à l'aide des fonction comme mkdir ou chdir ou grâce à la structure dirent et quelques vérifications .

Reprenons notre exemple de 10 processus et 15 sauts, on a bien donc 10 répertoires (chacun numéroté) correspondant aux processus et dans ses répertoires on a bien le nombre de fichiers qui correspondant au nombre de visite que réalise le jeton.





Les signaux


UTILISATEUR
SIGUSR1
Père
Signaux
Application de Sigstop et sigCont avec la fonction kill()
jeton
Repertoire id2
visite1
visite2
...
id1
idn
id2
id3
id4
id5

Conclusion

Pour conclure , on aimerai préciser que la partie signaux n’a pas été réaliser car on a pas abouti à ce qu’on a voulu faire ..

On aimerai rajouter aussi qu’on pourrait penser à une nouvelle optimisation comme élire un processus fils afin qu’il est soit disant le délégué et donc qui pour discuter avec le processus père par signaux et donc donné toute information provenant du père aux autres processus fils.