Projet Systèmes Concurrents

Pierre Durollet

Janvier 2025

Contents

1	Introduction	1
2	Prise en main	1
3	Implantation	2
4	Résultats	2
5	Pistes Eventuelles	3

1 Introduction

Le but de ce projet est d'utiliser "ManuX" afin de voir la mise en place et l'effet de tubes en fonction de s'ils sont bloquants ou non.

2 Prise en main

Commandes intra interface Afin de comprendre l'environnement qui nous est donné, j'ai essayé, une fois le code lancé, et sur la console qui donne les étapes de l'initialisation du noyau ManuX, différentes touches afin de voir leurs effets, là on découvre l'effet de la touche "echap" qui change de console, mais egalement d'autres touches, qui sont résumées par la touche "h".

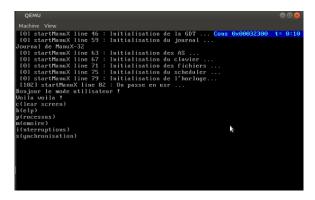


Figure 1: Commandes disponibles

Variables buffer et lecteurs/rédacteurs J'essaie également de modifier, dans le fichier init-acces-concurrent.c les valeurs de tailles des buffers et du nombre de lecteurs/rédacteurs afin de correctement voir ce que cela change. Ainsi on voit effectivement des changements, pour la taille du buffer, cela fait varier la quantité de caractères lu d'un coup, de la taille indiquée moins 1. En faisant varier le nombre de lecteurs/rédacteurs, c'est assez explicite, cela va changer le nombre de fenêtres de lecteurs et de rédacteurs en fonction de la valeur mise.

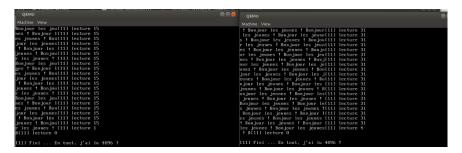


Figure 2: Comparaison entre un buffer de taille 16 et un de taille 32

3 Implantation

Le but de ce projet n'étant pas uniquement de découvrir des fonctions déjà codées, j'ai commencé les changements du code en ajoutant une variable *ExclusionMutuelle* dans la définition du tube dans *tubes.c.* Cette variable est importante afin d'empêcher l'entrée dans un tube alors qu'on ne veut pas, ce qui corresponds au cas bloquant. Bien évidemment, la variable seule n'est pas suffisante, il faut également modifier les méthodes des tubes afin de l'utiliser correctement. C'est ainsi que les méthodes *tubeOuvrir*, *tubeFermer*, *tubeEcrire et tubeLire* ont été modifiées à cet effet.

J'ajoute également deux variables d'état sur l'état du tube, permettant d'indiquer si il est plein ou si il est vide. Ces variables interviennent dans la condition d'attente. Effectivement, si le tube est plein, on ne peut pas y rentrer.

Je rajoute un printf() juste après l'ordonnanceur dans la méthode tubeLire afin de voir clairement voir où est mis l'ordonnanceur.

4 Résultats

Ainsi, on voit que la lecture se fait bien, se termine bien, et affiche également le message de fin.

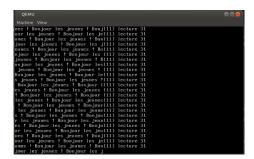


Figure 3: Lecture en cours

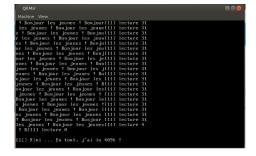


Figure 4: Lecture finie

5 Pistes Eventuelles

On peut essayer de créer un 2e écrivain qui va écrire autre chose afin de voir en détail les changements et les liens entre quel lecteur a lu après quel écrivain. Ou de faire afficher le nombre de lecteurs/rédacteurs en attente lorsqu'un fini.