Travaux Pratiques & Travaux Dirigés
Temps-Réel et Parallélisme

Durée : 6h.

Objectifs

Les objectifs de ce TP sont multiples :

- concevoir une application multitâche (non temps réel) avec une approche dirigée par les événements.
- manipuler les mécanismes d'un système d'exploitation dit « asynchrone » pour implémenter cette application.
- concevoir cette même application avec une approche dirigée par le temps
- utiliser les mécanismes d'une chaine de production dirigée par le temps et orientée sûreté de fonctionnement pour implémenter cette application
- synthétiser les avantages/inconvénients de chacune des approches.
- Aborder les problèmes d'interblocage et de famine dans les approches par événements

Note : Nous ne cherchons pas à faire dans ce TP une application temps réel. Ainsi, pour l'approche dirigée par les événements, nous allons utiliser un système d'exploitation classique Unix et les apis normatives POSIX. Les apis que vous allez manipuler, les phénomènes que vous allez observer et la conception que vous allez mettre en œuvre dans ce TP seraient reproductibles sur un système d'exploitation dit « Temps Réel ».

Sources

Le code source pour ce tp est présent à l'adresse suivante :

https://github.com/fthomasfr/multitasking_training_practical_work

Pour le récupérer vous pouvez le télécharger directement ou utiliser git :

```
git clone https://github.com/fthomasfr/multitasking_training_practical_work.git
```

L'ensemble des informations et codes d'exemples concernant la programmation multitâche avec les apis POSIX sont disponibles en ligne: http://pubs.opengroup.org/onlinepubs/9699919799/
L'ensemble des informations et codes d'exemples concernant la programmation Psy dans Asterios Developer sont disponibles sur dans le logiciel Asterios directement :

- Aide -> Documentation -> User Guide décrit l'utilisation des moyens de communications et l'utilisation du simulateur.
- Aide -> Documentation -> PsyC Reference Guide décrit l'ensemble des mots clés du PsyC

TP 1ere Partie

Nous devons réaliser un logiciel sûr de fonctionnement satisfaisant la spécification et l'architecture suivante :

- Notre logiciel possède capture quatre entrées numériques asynchrones, une sortie numérique et une sortie de diagnostic. Une entrée est caractérisée par un tableau de 256 entiers. La sortie numérique est caractérisée par un tableau de 256 entiers. La sortie diagnostic est une chaine de caractères à destination du terminal.
- **Exigence 1 :** Notre logiciel doit acquérir quatre entrées et produire en sortie le cumul des entrées. Le cumul correspond à la production d'un tableau de 256 entiers dont le i^{eme} élément de ce tableau est la somme des i^{emes} éléments des tableaux d'entrée.
- **Exigence 2 :** Notre logiciel doit acquérir toutes les données d'entrées.
- **Contrainte de conception 1 :** Notre logiciel doit garantir que les données d'entrée sont correctement formées avant de les sommer.
- **Exigence 3 :** Notre logiciel doit produire sur la sortie diagnostic pour chaque opération de cumul, combien d'entrées ont été acquises, combien ont été sommées et combien restent à sommer.

Conception en utilisant une approche dirigée par les événements

Plusieurs conceptions peuvent exister. Nous proposons de guider cette conception en utilisant les mécanismes suivants:

- Utilisation de tâches pour la capture de données.
- La mémoire est partagée entre les tâches.
- Partage d'un tableau de messages. Un message est une entrée acquise.
- Utilisation d'événements sans transmission de données pour synchroniser les tâches
- Utilisation d'un checksum pour satisfaire la **Contrainte 1**
- L'utilisation d'une librairie de messages fournis. La fonction MessageFill permet de simuler la production d'une entrée. La fonction MessageDisplay permet d'afficher le message.

1. Etudiez la librairie de messages fournis (fichier msg.h et msg.c) pour en comprendre les structures de données et les apis.
    a. Pourquoi certaines variables sont-elles considérées comme des variables C volatiles ?
2. Ecrivez un chapitre d'architecture logiciel (Software Architecture Document) allouant les exigences précédentes. Cette conception devra utiliser des diagrammes de composants pour décrire l'architecture structurelle et des diagrammes de séquences pour expliquer et démontrer la causalité des traitements (architecture comportementale).
    a. Pour cela nous utiliserons https://www.draw.io et la librairie UML
3. Nous avons orienté la conception par l'utilisation des tâches. Nous aurions pu utiliser des processus. Citez les caractéristiques intéressantes des processus pour une conception orientée « sûreté de fonctionnement » ? Citez également celles qui ne sont pas satisfaites ?

Implémentation dirigée par les événements

4. Implémentez votre conception (Implémentation + Exécution)
5. En quoi votre conception et votre implémentation ne satisfait pas les exigences de sûreté de fonctionnement vues en cours ?

Pour aller plus loin en conception et implémentation dirigées par les événements

6. Comment auriez-vous pu protéger de manière efficace, sûr, sans mutex et sans sémaphores les compteurs permettant d'indiquer pour chaque consommation, combien de messages ont été produits, combien ont été consommés et combien reste à consommer ?
    a. Pour cela réfléchissez à ce que le compilateur et le processeur peuvent nativement proposer
    b. Le compilateur GCC donne accès ainsi à une série de fonction dans https://gcc.gnu.org/onlinedocs/gcc-4.1.0/gcc/Atomic-Builtins.html#Atomic-Builtins Utiliser dans votre programme certaines de ces APIs.

Conception en utilisant une approche dirigée par le temps

7. Proposez un chapitre d'architecture détaillée en utilisant le Psy satisfaisant l'ensemble des exigences énoncées dans la section précédente et utilisant les directives ci-dessus. Cette conception devra utiliser un diagramme temporel pour expliquer et démontrer la causalité des traitements.
8. Quels sont les gains d'une approche dirigée par le temps pour ce programme ?
    a. Quels sont les propriétés temps réel que vous êtes en mesure de démontrer dès la conception ? Auriez-vous pu le faire dans l'approche précédente ?
    b. D'autres propriétés ?

Implémentation dirigée par le temps

Nous avons réalisé un squelette de programme Psy (L'objectif n'est pas de vous apprendre les APIs psy) dans le fichier mySoftwarePsyDesign.psy.

9. Implémentez votre conception (Implémentation+Execution)
10. Expliquez pourquoi le compte de production ne peut pas se faire de la même manière que dans l'approche asynchrone précédente ?
11. Comment pourriez-vous concevoir votre programme pour permettre ce passage d'information ?
12. Est-ce que nous aurions pu passer par pointeur pour transmettre cette information ?
    a. Implémentez en Psy le passage de cette information entre le producteur et le consommateur par pointeur.
    b. Quel mécanisme est déclenché avec cette dernière implémentation ?

TP 2ieme Partie : LE DÎNER DES PHILOSOPHES

Exigences à satisfaire

« Il y a très longtemps, un riche philanthrope fonda un collège pour y accueillir d'éminents philosophes. Chacun d'eux disposait d'une pièce où il pouvait s'adonner à son activité professionnelle : penser. Il y avait aussi une salle à manger commune, dotée d'une table circulaire et d'autant de chaises que de philosophes présents. A la gauche de chaque place, était posée une fourchette d'or, tandis qu'au centre de la table trônait un grand plat de spaghettis, continuellement approvisionné.
Un philosophe devait passer le plus clair de son temps à penser, mais, quand il avait faim, il venait dans la salle à manger, s'asseyait à sa place, prenait sa propre fourchette à sa gauche et la plongeait dans le plat. Mais les spaghettis sont d'une nature si récalcitrante qu'il faut une deuxième fourchette pour les amener jusqu'à la bouche. Le philosophe devait donc aussi prendre la fourchette située à sa droite. Quand il avait fini, il devait poser ses deux fourchettes, se lever et retourner penser. Bien sûr, une fourchette ne pouvait servir qu'à un seul philosophe à la fois. Celui qui la voulait devait attendre qu'elle soit libre. »

On suppose qu'il y a au plus 5 philosophes qui peuvent être assis à la table de repas du collège.
On considère qu'un processus principal gestionnaire est chargé de mettre en place initialement et de détruire l'environnement d'exécution (e.g. les sémaphores).

A chaque philosophe est associé une même tâche fille : chaque tâche philosophe possède le même code exécutable lancé depuis le processus père.
Le processus philosophe est contrôlé par l'utilisateur. Un philosophe peut soit penser, soit être à table, soit manger.

Lorsque le processus est lancé, par défaut, le philosophe pense. Il peut alors choisir entre manger ou quitter (par une séquence prédéfinie dans le programme).

Lorsque la séquence demande à ce que le philosophe aille manger alors qu'il est en train de penser, le programme doit alors choisir un numéro de place à la table (les numéros vont de 0 à 4). Le philosophe prend alors sa propre fourchette à sa gauche puis celle à sa droite. Pour chaque saisie de fourchette, si elle est déjà prise, le philosophe doit attendre qu'elle soit libérée par son détenteur avant de continuer.

Enfin, lorsqu'un philosophe est en train de manger, il peut quitter la table pour retourner penser.

Travail à réaliser

1. Quelles sont les ressources partagées qui synchronisent le repas des philosophes ?

2. Concevez un programme en utilisant une approche par événement

3. Dans les hypothèses du travail à effectuer, mettez en évidence une configuration d'interblocage avec les 5 philosophes présents à chaque place de la table. Donnez le scénario qui permet d'atteindre cet interblocage.

4. Proposez une solution pour éviter l'interblocage (4).

5. Proposez une configuration (et un scénario associé) où un philosophe est en famine. Que peut-on faire pour l'éviter ?

6. Programmez en langage C et à l'aide des primitives systèmes offertes par WINDOWS le problème des philosophes ainsi proposé en 4. Votre solution sera constituée d'un seul programme : philosophe.c.

7. Pourriez-vous proposer une conception et une implémentation en PsyC ?

Bonus

8. Proposez une modélisation en un réseau de pétri pour montrer que votre conception à la question 5 ne comporte pas d'interblocage.

TP Bonus : DASH

Nous souhaitons réaliser un programme en PsyC dédié à l'affichage en console du patron suivant :
---\n
---\n
-\n
-\n

Il doit exister dans ce programme deux entités :

- Un afficheur du caractère «-» (Dash)
- Un gestionnaire du patron affichant le retour à la ligne (\n)

Les exigences sont les suivantes :

- Le patron doit être respecté et reproduit cycliquement
- Chaque caractère «-» doit être affiché tous les 400ms +/-200ms
- Une séparation spatiale et temporelle doit être effective entre les deux fonctions pour ne pas partager l'accès à la console (non atomicité du printf)
- Aucune synchronisation explicite ou communication de données ne doit exister entre les fonctions