TP MI03 - Réalisation d'un mini noyau temps réel (Parties 5 et 6).

5ème partie : le dîner des philosophes

Ce problème de synchronisation classique a été posé et résolu en 1965 par E. W. Dijkstra. Cinq philosophes sont réunis pour dîner et pour philosopher. Cinq couverts composés d'une assiette et d'une fourchette sont disposés autour d'une table circulaire. Chaque convive a devant lui un plat de spaghettis tellement glissants qu'il faut deux fourchettes pour les manger. Après une longue réflexion les philosophes décident du protocole suivant :

- Chaque philosophe reste à une place fixe
- Pour manger, un philosophe ne peut emprunter que la fourchette de droite et de gauche
- Initialement tous les philosophes pensent
- Un philosophe mange durant un temps fini
- Un philosophe ne peut être que dans une des trois situations suivantes :
    - Il pense pendant un temps indéterminé,
    - Il a faim,
    - Il mange.

On vous demande de proposer une solution dans laquelle chaque philosophe est simulé par une tâche distincte et qui permet à deux philosophes de manger en même temps, tout en évitant les interblocages.

6ème partie : Communication par tubes

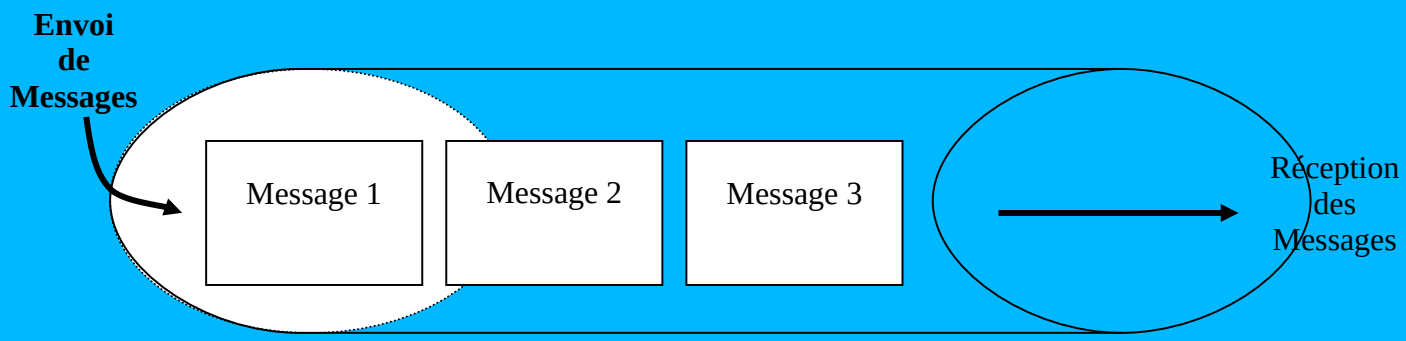
Pour l'instant la gestion de communications entre les tâches Producteur / Consommateur est gérée par le programmeur avec l'utilisation de variables globales. La fréquence d'utilisation de ce modèle de communication incite à réaliser, comme pour la gestion des sémaphores, des **mécanismes d'échange de plus haut niveau**. Ces mécanismes doivent permettre l'échange de données entre tâches (et même entre machines distantes) en évitant au programmeur les détails de l'exclusion mutuelle et de la synchronisation.

Boîtes aux lettres et tubes

Les boîtes aux lettres (*mailboxes*) et les tubes (*pipes*) sont deux mécanismes de communication semblables. Dans le cas des boîtes à lettres, on utilise une zone de mémoires tampons de taille fixe. Les messages envoyés ou lus sont donc de taille fixe, et la taille est définie à la création de la boîte aux lettres.
Une variante de la boite aux lettres est le tube. Dans ce cas la taille de lecture ou d'écriture des messages est variable. Dans les deux cas il s'agit de mécanismes de communication **unidirectionnels**.

Ex : l'écriture de 10 messages de 1 octet peut être lue comme 1 message de 10 octets.

Dans ces deux cas la mémoire est de type FIFO.



On se propose dans ce TP de réaliser un mécanisme de communication par tubes. Vous regrouperez les déclarations dans le fichier *PIPE.H* et les implémentations dans le fichier *PIPE.C*.

Cahier des charges

- Taille variable des messages
- Suite d'octets organisée en FIFO
- Pas de séparation entre les messages
- La quantité de données à lire ou écrire sont spécifiée dans les primitives.
- Si le lecteur accède à un tube vide, il se bloque.
- Si le tube est plein, l'écrivain se bloque.
- Ils sont débloqués par une action de leur vis a vis.
- Les tubes sont privés aux tâches qui leurs sont associées.

Primitives de gestion d'un tube et structures de données associées

Primitives

- Allocation du conduit : *unsigned p_open(unsigned redacteur, unsigned lecteur)*
- Libération du tube : *void p_close (unsigned conduit)*
- Lecture dans un tube : *void p_read (tube, donnees, quantité)*
- Ecriture dans un tube : *void p_write(tube, donnees, quantité)*

Structures de données utilisées

```
typedef struct {
        ushort pr_w , pr_r ;        /* redacteur & lecteur du tube */
        ushort ocupp ;              /* donnees restantes */
        uchar is, ie ;              /* pointeurs d'entree / sortie */
        uchar tube[SIZE_PIPE] ;     /* Tampon */
} PIPE;

PIPE  _pipe[MAX_PIPES] ;            /* Variables tubes */
```

Initialisation des tubes

- Mettre *pr_w = MAX_TACHE*, ce qui représente le fait que tous les tubes sont inutilisés.

Création d'un tube

- Vérifier que les taches sont bien créées
- Vérifier qu'il n'existe pas de tube avec ces 2 taches
- Trouver un tube non utilisé
- Initialiser le tube :
- *pr_w = redacteur*
- *pr_r = lecteur*
- *is = ie = 0*
- Retourner le numéro du tube créé

Ecriture dans un tube

- Vérifier que le tube existe et que la tâche en est propriétaire
- Vérifier qu'il y a de la place dans le tampon, sinon endormir la tache
- Si le tube est vide et si la tache lectrice est suspendue **sur une lecture de ce tube** alors la réveiller
- Copier les données dans le tampon

Lecture dans un tube

- Vérifier que le tube existe et que la tâche en est propriétaire
- Vérifier que le tampon n'est pas vide, sinon endormir la tache
- Si le tube est plein et si la tâche écrivain est suspendue **sur une écriture dans ce tube** alors la réveiller
- Copier les données à partir du tampon.