

Concurrence

(programmation concurrente)

Objectifs d'enseignement

- définir les notions de processus, de synchronisation et de communication entre processus;
- présenter et résoudre quelques problèmes types liés à la gestion de la concurrence;
- illustrer quelques solutions avec des instruments de programmation auxquels les étudiants ont facilement accès.

Objectifs d'apprentissage

- reconnaître la nature d'un problème rencontré;
- savoir élaborer une démarche visant à le résoudre;
- connaître les ressources techniques que l'on peut engager pour ce faire, leurs caractéristiques et leurs propriétés;
- savoir les mettre en œuvre;

dans le domaine des systèmes concurrents.

Supports de cours

- Programmation concurrente:
[A. Schiper](#)
[Programmation Concurrente, PPUR](#)
- « fichier.pdf » des notes présentées en cours
(mis à disposition par les assistants).

Calendrier

- | | | | |
|-----------|--------------------|-----------|----------------------------|
| • lu 7.3 | Cours | • lu 2.5 | Cours |
| • lu 14.3 | Exercices | • lu 9.5 | Exercices |
| • lu 21.3 | Cours | • lu 16.5 | Pentecôte |
| • lu 28.3 | Vacances de Pâques | • lu 23.5 | Cours |
| • lu 4.4 | Cours | • lu 30.5 | Quiz / Exercices |
| • lu 11.4 | Exercices | • lu 6.6 | Cours |
| • lu 18.4 | Cours | • lu 13.6 | Exercices /
répétitoire |
| • lu 25.4 | Quiz / Exercices | | |

Programmation concurrente =
programmation parallèle sur un seul processeur

La programmation concurrente est liée au fonctionnement des entrées/sorties:

- les e/s peuvent se dérouler simultanément à l'activité du processeur;
- les e/s donnent lieu à des activités concurrentes.

- **Introduction**
- Entrées-sorties et interruptions → 7 mars
- Concept de processus
- Exclusion mutuelle → 21 mars
- Sémaphores → 4 avril
- Coopération entre processus
- Programmation concurrente en Java → 18 avril
- Moniteurs → 2 mai
- Ecriture d'un noyau en Modula-2 → 23 mai
- Threads Posix → 6 juin

Exemple: un atelier de menuiserie

Activité du menuisier	~	Exécution d'un programme sur un processeur
– téléphone		2 périphériques d'entrée
– guichet		
– tapis roulant		1 périphérique de sortie

Programmation séquentielle (du menuisier)

```
loop
  attendre une commande;
  exécuter la commande;
  déposer le bois sur le tapis roulant;
end loop;
```

Amélioration par l'introduction d'interruptions:

- **sonnerie au téléphone et au guichet;**
- **sonnerie au tapis roulant.**

Le menuisier doit s'organiser différemment:

- **carnet pour noter les commandes;**
- **étagère pour déposer le bois coupé.**

Inconvénients de cette solution:

- **du point de vue des clients: attente;**
- **du point de vue du menuisier: attente (si le tapis roulant est occupé).**

Programmation concurrente ad-hoc

- tant qu'il y a des commandes inscrites dans le carnet, exécuter la commande suivante et la déposer sur l'étagère;
- lorsqu'une sonnerie retentit à un point d'entrée, suspendre l'exécution, noter la commande, et reprendre l'exécution de la commande en cours;
- lorsque la sonnerie du tapis retentit, suspendre l'exécution en cours, prélever un lot sur l'étagère, le déposer sur le tapis, et reprendre l'exécution de la commande en cours.

Avantages:

- **les clients n'attendent plus aux points d'entrée;**
- **le menuisier n'est plus oisif en attendant que le tapis roulant se libère.**

Inconvénients:

- **la tâche du menuisier est plus complexe, ce qui est source d'erreurs.**

Problème:

- **l'étagère a une capacité limitée.**
-

Programmation du menuisier:

```

loop
  prendre le carnet;
  lire une commande;
  remettre le carnet;
  exécuter la commande;
  déposer la commande sur l'étagère;
end loop

```

Ce programme est **séquentiel** .

Programmation concurrente judicieuse

Introduire des préposés aux points d'entrée et de sortie:

- **préposé au téléphone;**
- **préposé au guichet;**
- **préposé au tapis roulant.**

Préposés et menuisier = *processus* ou *threads*

Total: 4 processus

Programmation du préposé au téléphone (resp. au guichet):

```

loop
  attendre la sonnerie du téléphone;
  prendre le carnet;
  noter la commande;
  remettre le carnet;
end loop

```

Ce programme est **séquentiel** .

Programmation du préposé au tapis roulant:

```

loop
  attendre que l'étagère soit non vide;
  prendre un lot;
  attendre la sonnerie du tapis roulant;
  déposer le lot sur le tapis roulant;
end loop

```

Ce programme également est **séquentiel** !

Menuisier:

```

loop
  prendre le carnet;
  lire une commande;
  remettre le carnet;
  exécuter la commande;
  déposer la commande sur l'étagère;
end loop

```

Préposé au téléphone:

```

loop
  attendre la sonnerie du téléphone;
  prendre le carnet;
  noter la commande;
  remettre le carnet;
end loop

```

Préposé au tapis roulant:

```

loop
  attendre que l'étagère soit non vide;
  prendre un lot;
  attendre la sonnerie du tapis roulant;
  déposer le lot sur le tapis roulant;
end loop

```

Préposé au guichet:

```

loop
  attendre la sonnerie au guichet;
  prendre le carnet;
  noter la commande;
  remettre le carnet;
end loop

```

Observation:

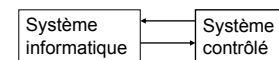
grâce à la notion de *processus* (menuisier, préposés) on a transformé la programmation d'une activité complexe en 4 activités séquentielles.

On a aussi introduit de nouveaux problèmes:

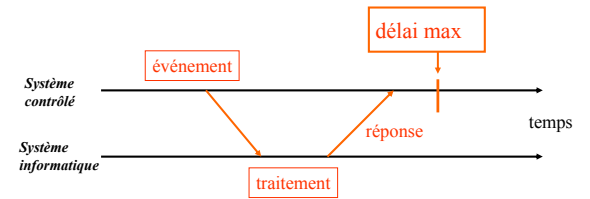
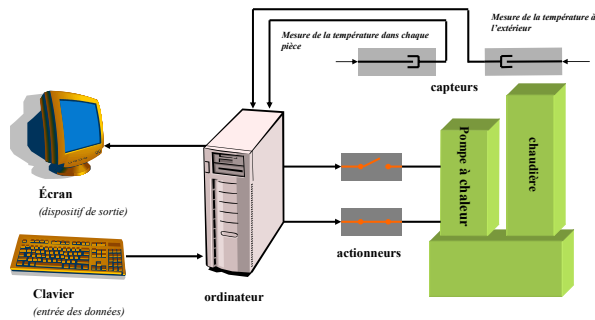
- gérer le partage du carnet entre le menuisier et les 2 préposés aux points d'entrée (*exclusion mutuelle*);
- synchroniser le menuisier et le préposé au tapis (attente si l'étagère est pleine, resp. vide).

Programmation concurrente et programmation en temps réel

- un programme en temps réel, tout comme un programme concurrent, gère des périphériques;
- les techniques de la programmation concurrente interviennent dans la programmation temps réel
- **en plus, un programme temps réel contrôle un ou plusieurs systèmes externes → contraintes de temps**



- robots industriels;
- commandes de chauffage d'immeuble;
- contrôle d'expériences de laboratoire;
- contrôle de navigation aérienne.



on distingue

- temps réel avec échéance lâches: *(soft) real time*;
- temps réel avec échéances strictes: *hard real time*.

L'échéance est l'instant où un traitement doit être terminé.



pour l'industrie:

