Informatique industrielle A7-19571
Systèmes temps-réel

J.F.Peyre

Partie I : Introduction

Plan de la première partie

- Quelques définitions
- Caractéristiques communes des applications temps-réel
- Exemples d'applications temps-réel
- Le choix d'un langage de développement
- Le choix d'un système ou d'un exécutif temps-réel
- Conclusion

Définition

- Un système temps réel est un système (application ou ensemble d'applications) informatique qui doit répondre à des stimuli fournis par un environnement externe afin de le contrôler
- La correction d'un tel système dépend non seulement de la justesse des calculs mais aussi du temps auquel est fourni la réponse (contraintes temporelles)
- Les ressources utilisées pour mener à bien les calculs et sont en nombre limité (contraintes matérielles)

Classification

- **Temps réel dur (hard real-time) : le non respect des contraintes temporelles entraîne la faute du système**
  - e.g. contrôle de trafic aérien, système de conduite de missile, ...
- **Temps réel souple (soft real-time) : le respect des échéances est important mais le non respect des échéance ne peut occasionner de graves conséquences**
  - e.g. système d'acquisition de données pour affichage
- **Temps réel ferme (firm real-time) : temps réel souple mais où il n'y a aucun intérêt à avoir du retard ou temps réel dur pour lequel quelques échéances peuvent être occasionnellement manquées**
  - e.g. projection vidéo

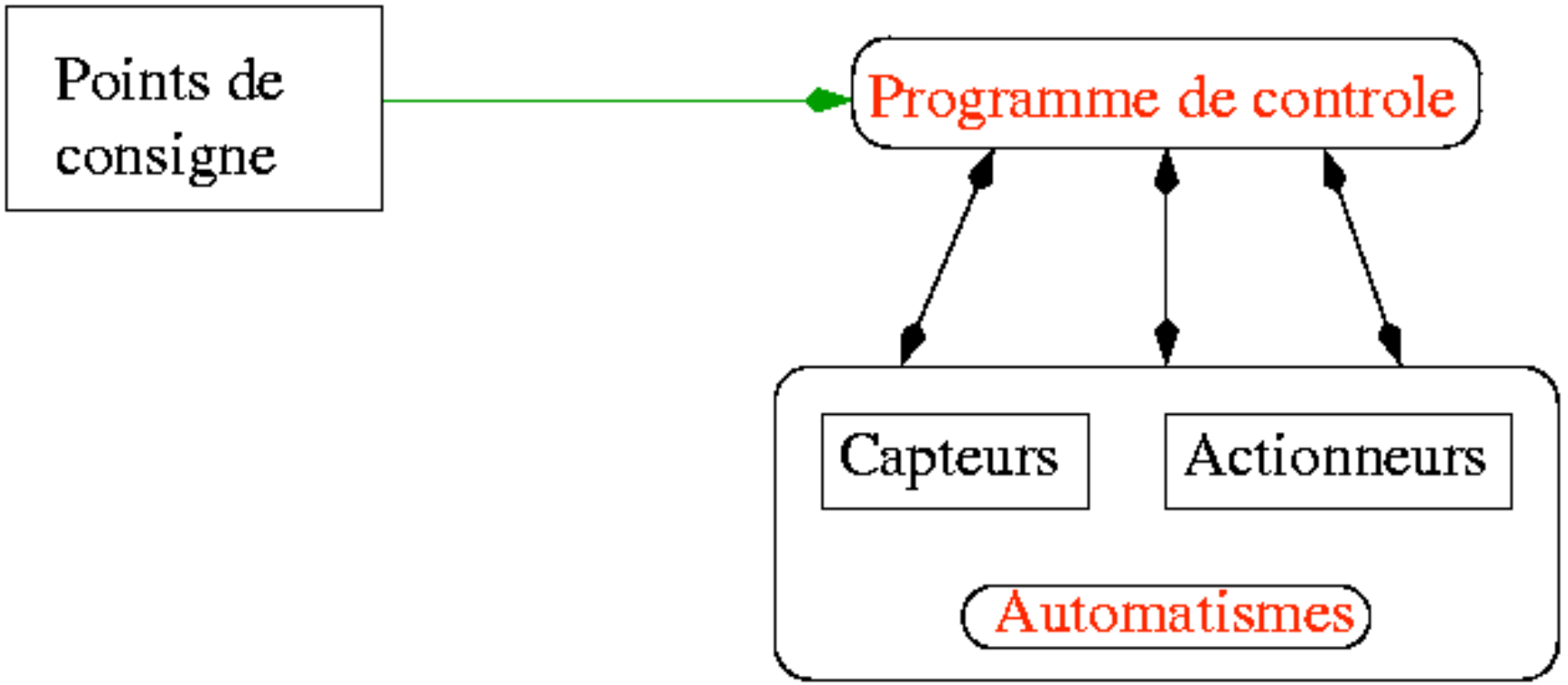
Classification (suite)

- Il ne faut pas confondre contrainte temporelle (qui dépend de l'application et de l'environnement) et rapidité de traitement (qui dépend de la technologie utilisée)

- Un système temps-réel inclut généralement différents sous-systèmes chacun pouvant avoir ses propres contraintes temporelles (dures, souples ou fermes)

Structure : Boucle ouverte

Structure : Boucle fermée



Caractéristiques communes des applications temps-réel

Large, complexe et fiable

- Un système temps réel interagit avec un environnement extérieur souvent complexe et en évolution
- Il doit pouvoir interagir avec différents types d'éléments matériels
- Il doit respecter des échéances temporelle
- Il doit garantir une fiabilité importante

Utilisation du temps concret

- **Au sein d'une application ou d'un système temps-réel if faut pouvoir manipuler le temps concret (horloge)**

- **Le temps réel (ou temps concret) sera utilisé de plusieurs façons:**
  - Soit en définissant la date à laquelle une action doit être ***commencée***
  - Soit en définissant la date à laquelle une action doit être ***finie***

- **Il peut être nécessaire de pouvoir modifier ces paramètres en cours d'exécution et de pouvoir préciser les actions à prendre en cas de faute temporelle**

Découpé en tâches ou en processus concurrents

- **Dans le monde réel les périphériques et l'environnement du système évoluent simultanément (en parallèle ou concurrence)**

- **Si l'on veut réduire la complexité de conception et calquer fidèlement la réalité il faut s'appuyer sur de la programmation concurrente :**

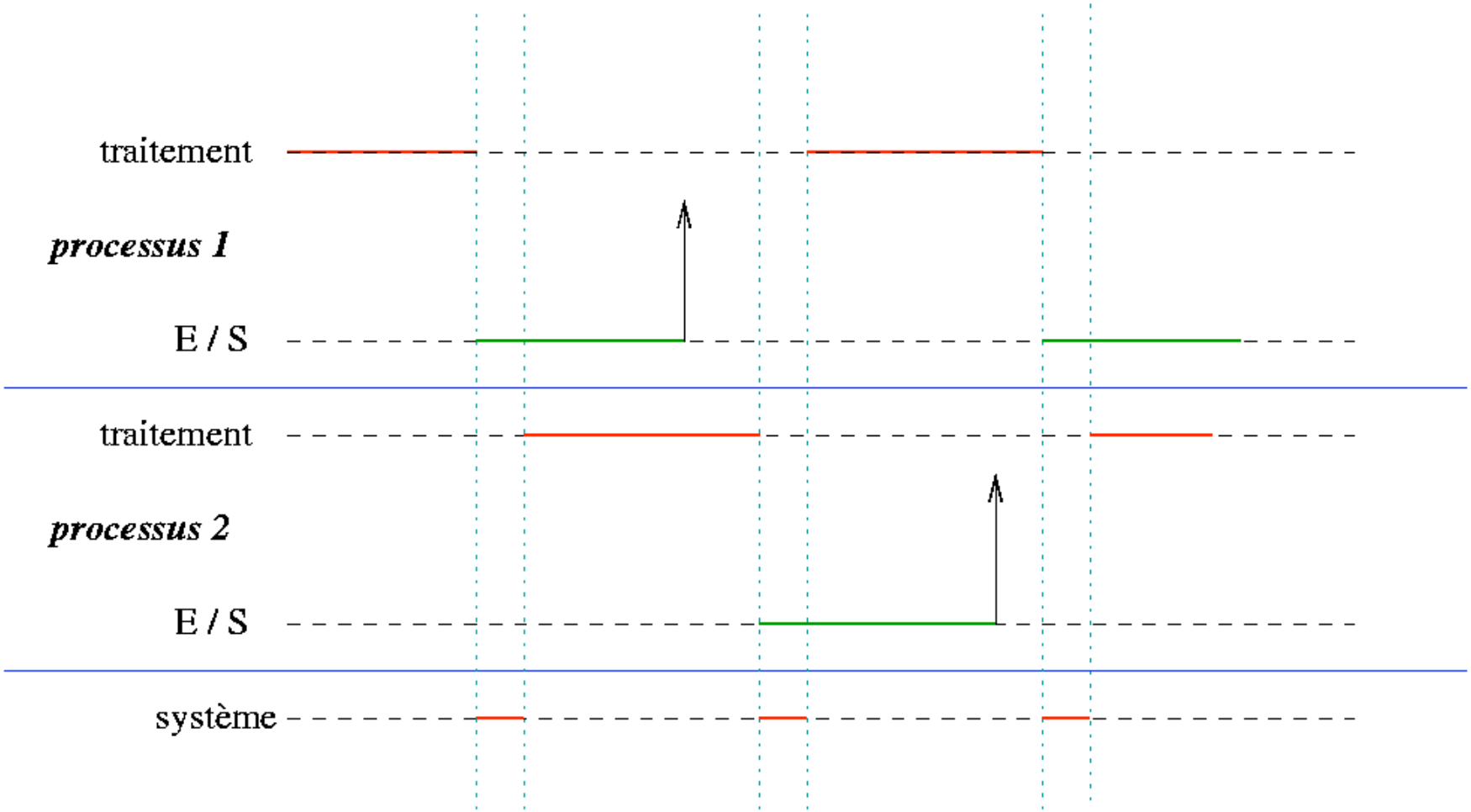
  – utiliser un modèle de tâches

  – utiliser des moyens de communication et de synchronisation inter tâches ou inter-process (mémoire partagée, boites aux lettres, files de messages, moniteurs, ...)

Découpé en tâches ou en processus concurrents (suite)

- Le modèle utilisé en programmation des systèmes temps réel est un modèle basé sur la concurrence (applications concurrentes)
- L'exécution de ces application se fait généralement dans un environnement mono-processeur
- On "simule" l'exécution concurrente des processus par la mise en ouvre du pseudo-parallélisme : le parallélisme est apparent à l'échelle de l'utilisateur mais le traitement sur le processeur (unique) est fait séquentiellement en tirant profit des entrées/sorties réalisées par les processus

Découpé en tâches ou en processus concurrents (suite)



Respect des échéances temporelles

- La limitation des ressources (en particulier du processeur) conduit à bloquer des processus (ils ne peuvent progresser du fait de manque de ressource)
- Afin de respecter en permanence les échéances, il faut gérer efficacement la pénurie et tenter de favoriser les processus dont l'avancement est le plus "urgent »
- Un ordonnancement consiste à définir un ordre sur l'utilisation des ressources du système afin de respecter les échéances temporelles

Respect des échéances temporelles (suite)

- **On appelle ordonnanceur (scheduler) le processus système qui gère l'ordonnancement des processus**

- **Un algorithme d'ordonnancement est une méthode ou stratégie utilisée pour ordonnancer les processus**

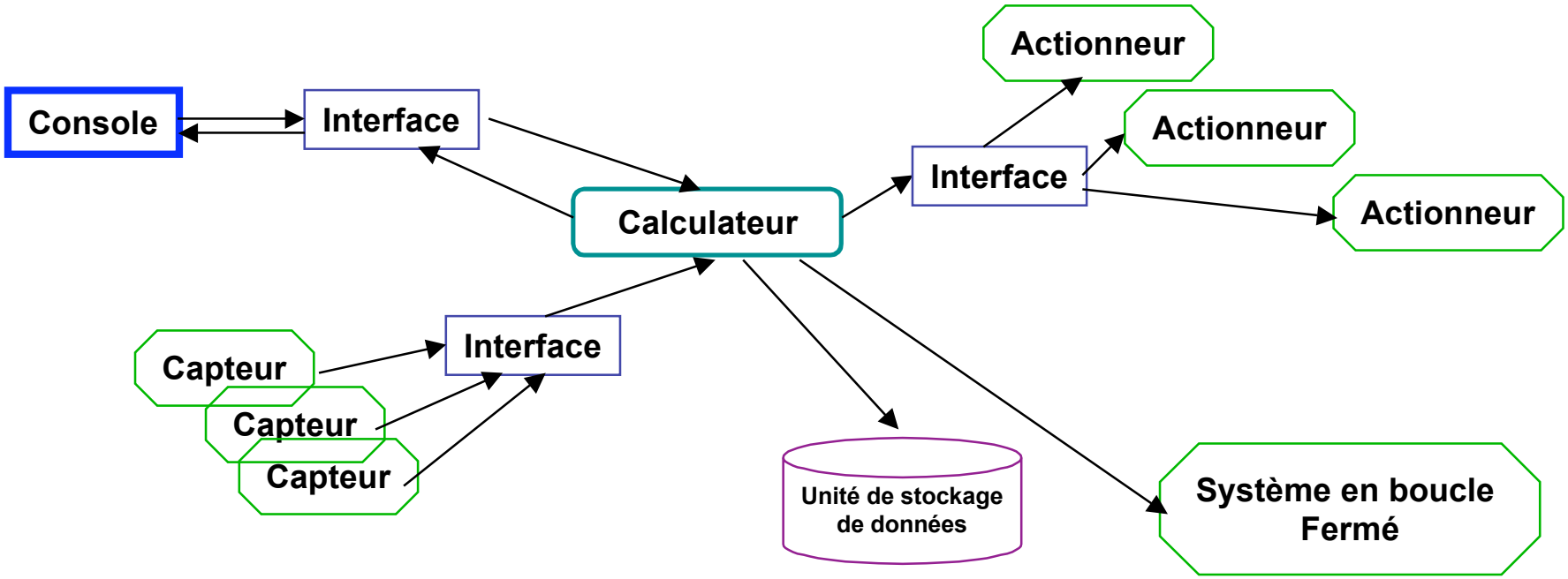
- **Un tel algorithme s'appuie sur la connaissance de certaines caractéristiques des processus ou du système**
  - procesus périodiques ou apériodiques;
  - processus cyclique ou non cyclique;
  - préemption possible ou pas;
  - échéance et pire temps d'exécution des processus
  - système à priorité fixe ou à échéance

Respect des échéances temporelles (suite)

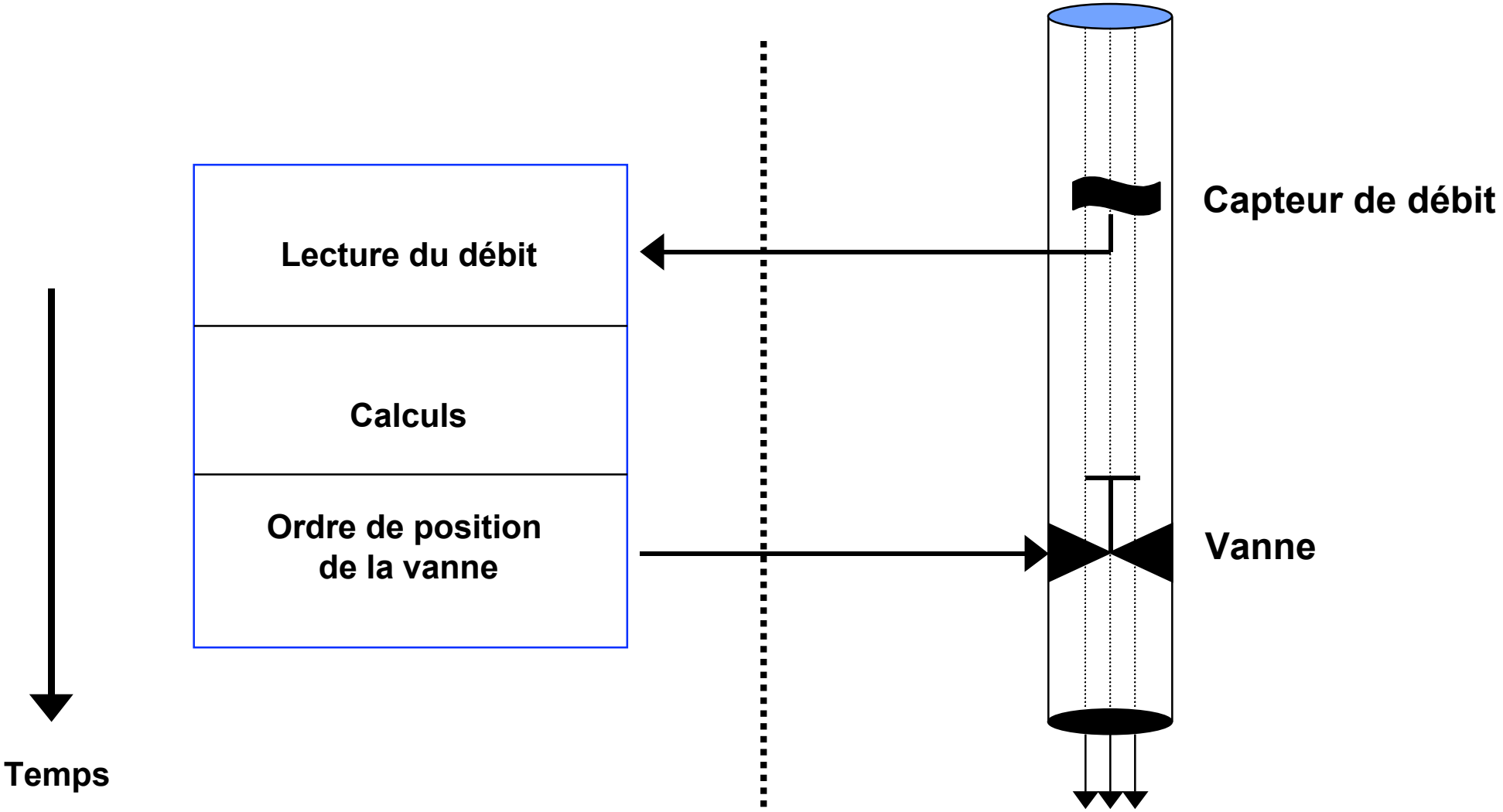
- **Deux algorithmes classiques d'ordonnancement**
  - RTM (RaTe Monotonic) : algorithme à priorité fixe pour processus cycliques (le processus le plus prioritaire est celui de plus petite échéance)
  - EDF (Earlest Deadline First) : algorithme à priorité dynamique pour processus cycliques (le processus le plus prioritaire est celui de plus petite échéance)
- **L'ordonnanceur choisit d'exécuter le processus prêt de plus haute priorité**
- **Au sein d'une même classe de priorité, le choix peut se faire par temps partagé (Round Robin) ou par ancienneté (gestion FIFO)**

Exemples d'applications temps-réel

Un système embarqué



Système de contrôle de débit



Le choix d'un langage de développement

Familles de langages possibles

- **Trois sortes de langages peuvent être identifié dans le contexte du développement d'applications temps réel**
  - les langages assembleurs
  - les langages séquentiel liés à des librairies système
  - les langages concurrents de haut niveau

Les langages de type « assembleur »

- **Historiquement, ces langages furent longtemps les seuls à être utilisés dans ce contexte**

- **Dépendant par nature de l'architecture cible (matériel et système d'exploitation)**

- **Aucune abstraction possible et grande difficulté de développement, de maintenance et d'évolution**

  -> Langages à proscrire sauf pour l'implémentation de petites fonctionnalités très spécifiques et apportant une grande amélioration des performances

Les langages séquentiels

- **Introduits pour remédier aux problèmes dûs au codage en assembleur**
- **Les plus connus sont le C, le C++ ou encore le Fortran**
- **Apporte un plus grand pouvoir d'abstraction et une certaine indépendance du matériel**
- **Mais, doit faire appel à des librairies systèmes spécifiques pour la manipulations des processus**

  -> Ces langages posent le problème de la standardisation des appels systèmes mais sont quelques fois le seul choix possible à cause de la spécificité d'une cible et des outils de développement sur celle-ci

Les langages concurrents

- **Langages généralistes incluant de plus la notion de tâches et des primitives de synchronisation**

- **Haut pouvoir d'abstraction, indépendance des architecture et des systèmes cibles (ou très peu dépendant)**

- **Parmi ces langages, Ada est sans doute le langage le plus abouti mais des restrictions de Java peuvent être utilisés à profit dans le développement d'applications/systèmes temps réel**

  -> Langages à privilégier lorsque d'autres contraintes (manque de formation, reprise de code existant, coopération inter-équipes/ ou inter entreprises, ...) ne rendent pas la chose impossible

Langages choisis dans ce cours

- Ada comme exemple de langage concurrent de haut niveau
- C avec l'utilisation de bibliothèques respectant la norme Posix 1003.1c (initialement 1003.4)

Le choix d'un système ou d'un exécutif temps réel

Système d'exploitation temps-réel

- **Caractéristique d'un système d'exploitation**
  - approche généraliste
  - supporte généralement plusieurs types d’applications simultanément
  - interaction par appels système
  - peu dépendant du domaine d'applications visé généralement de taille plus importante qu'un exécutif

Exécutif temps-réel

- **Caractéristique d'un exécutif**
  - système spécialisé
  - dédié à une application spécifique (système embarqué)
  - collection de primitives

-> plus spécialisé qu'un système d'exploitation code de taille plus petite qu'un système classique

Exemple de Système d'exploitation temps réel

- RT-Linux : basé sur du code libre, extension de Linux (http://fsmlabs.com/community/projects/)
- Lynx-OS : système Unix à base de thread noyau (http://www.lynuxworks.com/) compatible avec Linux
- QNX : système Unix (http://www.qnx.com/)
- Windows CE : système Microsoft temps réel (http://www.cewindows.net)

Exemple de d'exécutif temps réel

- VxWorks et pSos : Exécutif de Wind river (http://www.windriver.com)
- VRTX : http://www.mentor.com/vrtxos/
- mu-cos :

Conclusion

Les systèmes ou applications temps-réel sont

- Complexes
- Font intimement intervenir le temps dans leur conception
- Ont des besoins de fiabilité importants
- Généralement décomposés en sous-systèmes avec des tâches ou des processus qui interagissent
- Doivent être implémentés avec des langages appropriés
- Doivent être exécutés sur des systèmes ou des exécutifs adaptés

Vocabulaire *(petit Robert)*

- **Contrôler : (XX^{e}) maîtriser; dominer; avoir sous sa surveillance; soumettre à une régulation**

- **Contrôleur : appareil de réglage, de contrôle; activité de contrôle**

- **Concurrent : emprunté (1119) au latin *concurrens*, participe présent de *concurre* (-> concourir), proprement « *courir de manière à aller vers le même point* », « *se rejoindre* », puis employé en droit pour « *prétendre à la même chose en même temps* » avec l'idée de compétition; qui concourt au même but avec le sens néanmoins de compétition**

- **Concurrence : rivalité entre plusieurs personnes, plusieurs forces, poursuivant un même but;**