Systèmes d'Exploitation temps réel

Introduction aux STR

Gabriel FREY

2013

freyg.it@gmail.com

Plan de l'intervention

- 1. Introduction aux Systèmes Temps Réel
  - Définition & catégories
  - Notions importantes
  - Exécutif temps réel
  - Les STR dans les systèmes embarqués
  - Malentendus fréquents
  - Problèmes rencontrés
- 2. Programmation des STR
- 3. Ordonnancement
- 4. FreeRTOS

Définition

- John Stankovic (1988) : « En informatique temps réel, le comportement correct d'un système dépend, non seulement des résultats logiques des traitements, mais aussi du temps auquel les résultats sont produits ».

Définition

- Objectifs :
  - Déterminisme logique : les même données en entrée du système produisent toujours le même résultat .
  - Déterminisme temporel : le système doit respecter des contraintes temporelles (deadlines).
  - Fiabilité : le système répond à des contraintes de disponibilité.
- Un système temps réel est un système qui satisfait à des *contraintes temporelles*.

Définition

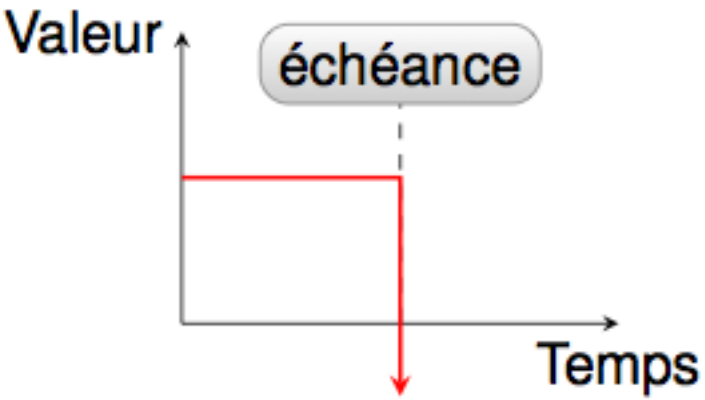
- Exemples de grandeurs de temps :
  - Milliseconde : systèmes radar
  - Seconde : temps de réponse des applications informatiques
  - Minute à quelques heures : systèmes de contrôle de production, contrôle du trafic
  - 24 heures : prévisions météo
  - Plusieurs mois, années : systèmes de navigation de sondes spatiales
- Alain Dorseuil and Pascal Pillot : « Le temps réel en milieu industriel », 1991.

Catégories

- 3 catégories de STR selon le respect des contraintes temporelles (degrés de tolérance) :
  - Temps réel « dur » (hard real time)
  - Temps réel « ferme » (firm real time)
  - Temps réel « mou » (soft real time)

Catégories

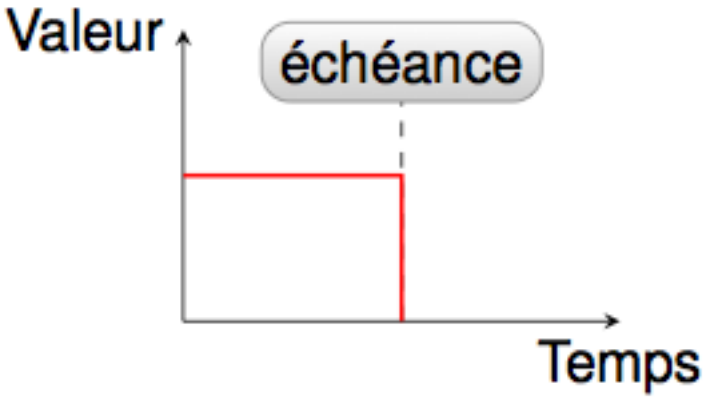
- Temps réel « dur » :
  - La réponse du système dans un temps imparti est *vitale*.
  - L'absence de réponse est *catastrophique*.
  - Exemples : contrôle d'une centrale nucléaire, ESP, systèmes embarqués utilisés dans l'aéronautique.



Catégories

- Temps réel « ferme » :
  - La réponse du système dans un temps imparti est *essentielle*.
  - Le résultat est *inutile* une fois la deadline passée.
  - Exemple : téléphonie.

- Assez subjectif !

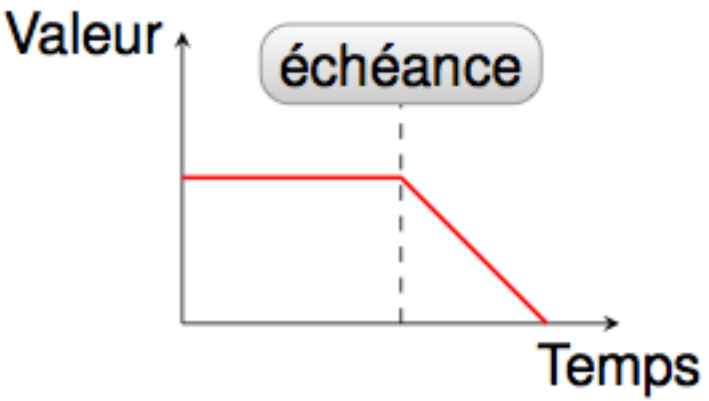


Catégories

- Temps réel « mou » :
  - La réponse tardive ou la non réponse du système n'a *pas de conséquences catastrophique* pour la vie du système.
  - Se rapprochent des systèmes d'exploitation classiques à *temps partagé.*
  - Exemples : multimédia ,

    système embarqué

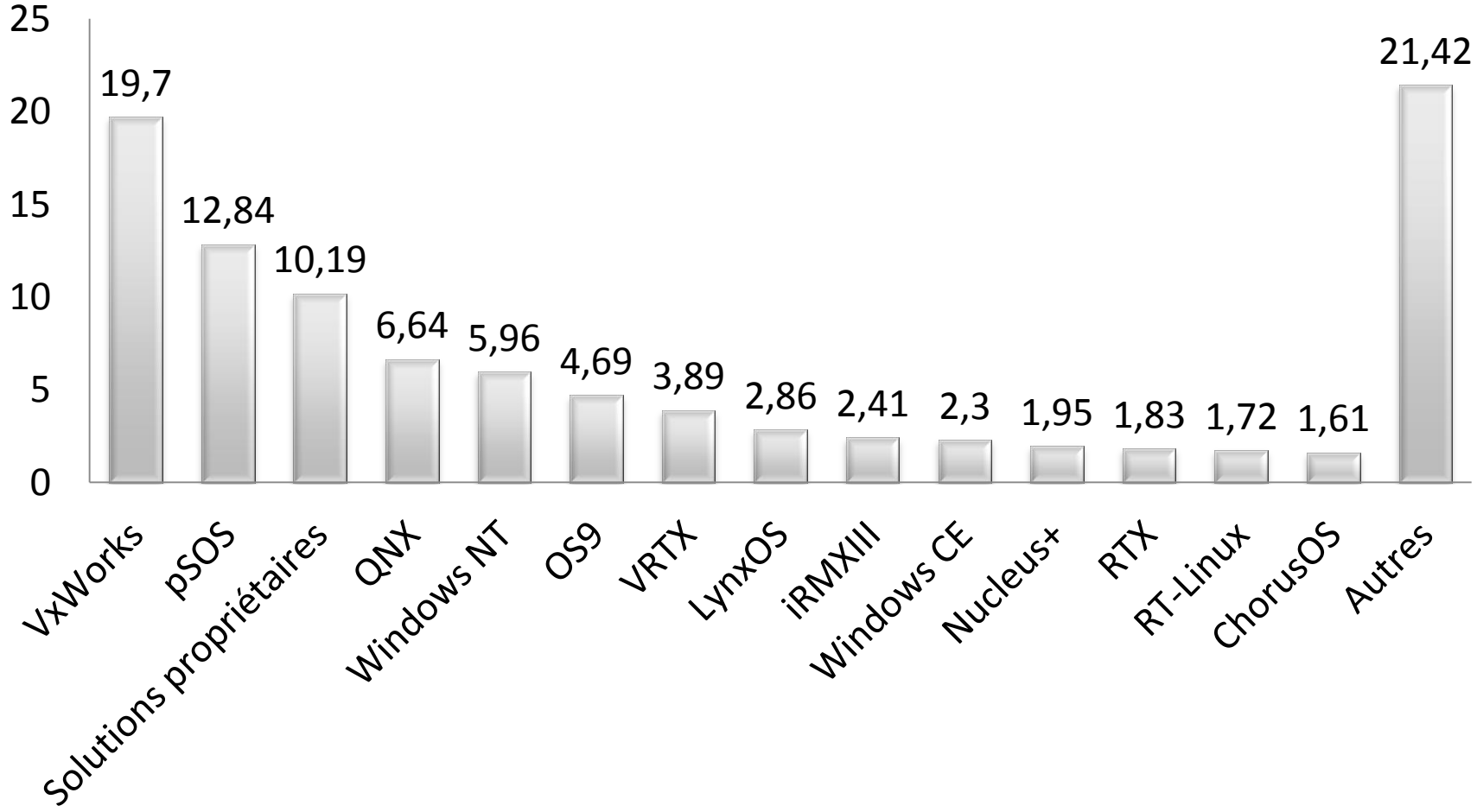
    sur téléphone, …



Catégories

- A retenir :
  - Tous les systèmes doivent répondre dans un temps imparti, c'est le *degré de tolérance* de non réponse qui caractérise les catégories.
  - Un même SE peut avoir des *sous-systèmes* TR « dur », « ferme », « mou ».

Marché des STR (%)

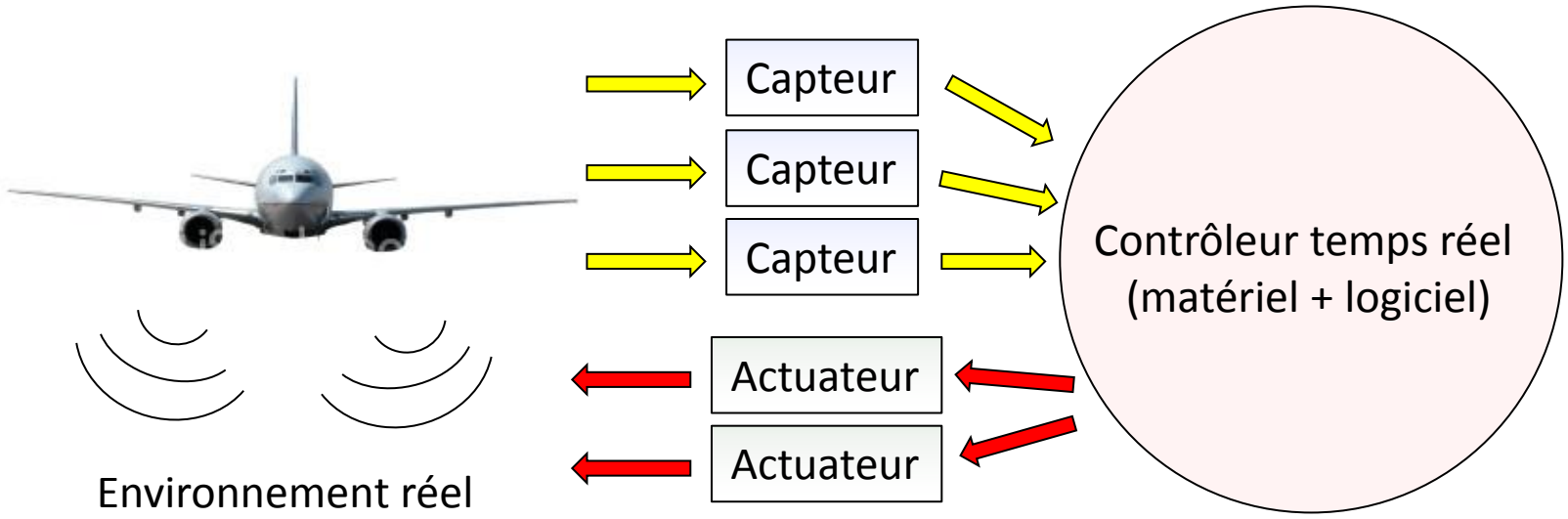


Marché des STR

- Diversité des produits : généralistes et très spécifiques. Importance des produits « maisons ».

- Exemples :
  - VxWorks : produit généraliste et largement répandu (Pathfinder, satellite CNES, …).
  - pSOS édité par ISI (appli militaire, tél. portable).
  - LynxOs (Unix temps réel).
  - Windows CE (systèmes embarqués peu temps réel).

Notions importantes



- *Système ouvert* (répond en continu aux sollicitations)

Notions importantes

- Prédictibilité : caractéristique *première* d'un système temps réel.
  - On cherche à déterminer à priori si le système va répondre aux exigences temporelles.
  - Le but étant de prouver que *toutes les contraintes (délais) sont respectées*, au moins pour les activités critiques.

- *Que préférez vous: un ordinateur de bord qui déclenche l'ABS à 1ms du blocage des roues dans 99% de cas, ou un qui le déclenche à 10ms, mais dans 100% de cas ?*

Notions importantes

- Concurrence : la tâche du système se décompose en plusieurs activités qui doivent souvent être exécutées *en parallèle*.
  - … car l'environnement est parallèle. Ex : quand il y a du vent, la gravitation ne s'arrête pas.
  - Activités plus ou moins critiques. Ex : ODB d'une voiture gère l'ESP, l'ABS, l'autoradio, ...

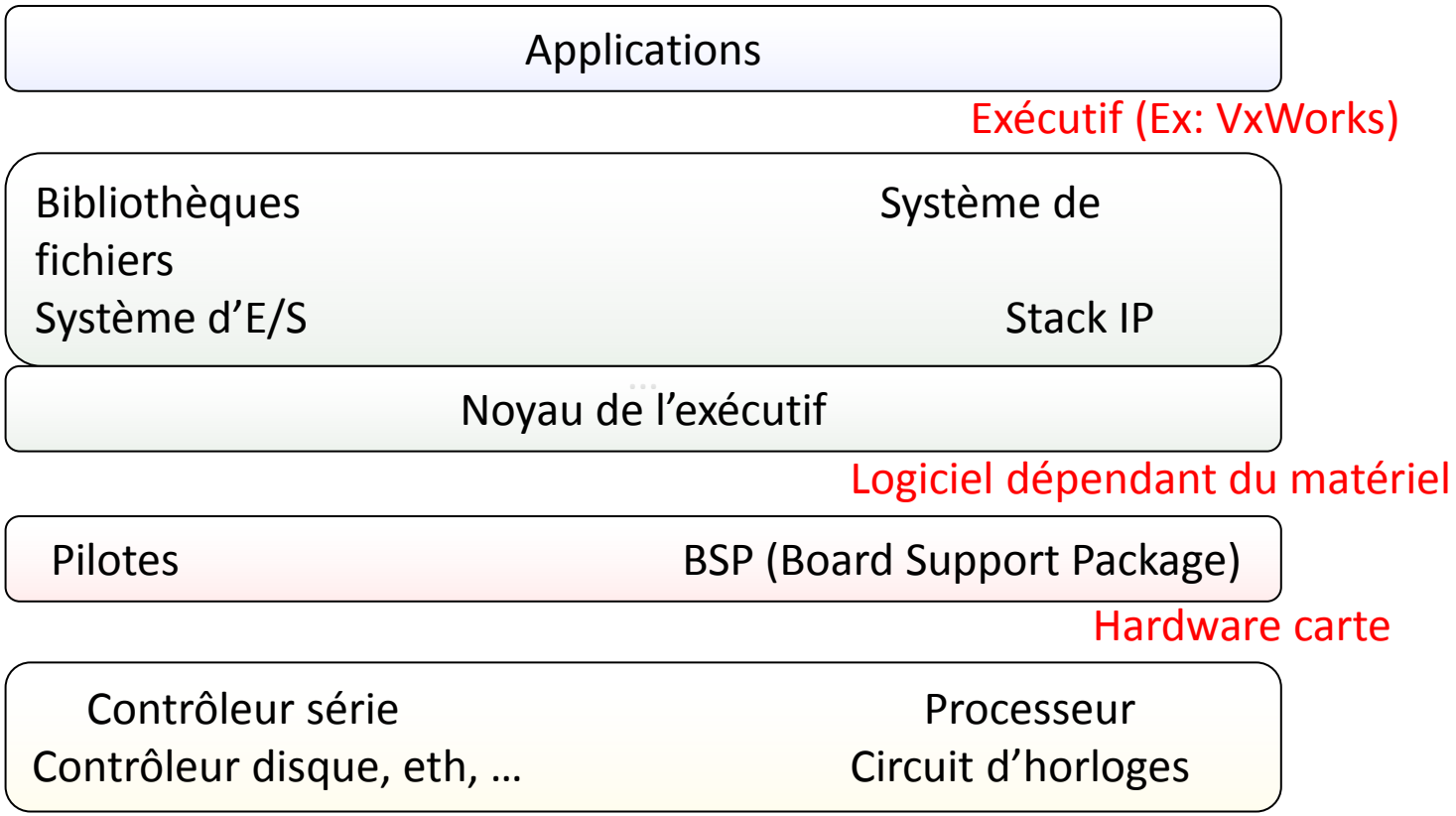
Notions importantes

- Préemption : capacité d'un système d'exploitation multitâche d'interrompre une tâche en cours en faveur d'une tâche de priorité supérieure.
- Un ordonnanceur préemptif peut interrompre une tâche au profit d'une tâche plus prioritaire.
- Un ordonnanceur non préemptif n'arrête pas l'exécution de la tâche courante.

Exécutif temps réel

- Caractéristiques :
  - *Flexible* vis à vis des applications.
  - Accès aisé aux ressources physiques.
  - Support pour langage temps réel (C, Ada).
  - Livré avec des *performances temporelles.*

Exécutif temps réel



Exécutif temps réel

- Performance déterministe et connue :
  - Utilisation de *benchmarks* : Rhealstone, Hartstone
- Critères de performance :
  - Latence sur interruption
  - Latence sur commutation de contexte/tâche
  - Latence sur préemption
  - Sémaphore « shuffle »
  - Temps de réponse pour chaque service (appel système, fonctions de bibliothèque).

Les systèmes embarqués

- Environ 99% des processeurs produits dans le monde sont destinés aux systèmes embarqués.

Les systèmes embarqués

- Contraintes supplémentaires :
  - Haute disponibilité requise : reset, update, … coûteux.
  - *Capacité* prédéterminée et limitée (calcul, stockage) -> limites variables.
  - Autres contraintes : consommation, effet Joule.

  (exemple : télescope Herschel)

- => Processeurs et couches système *personnalisés*.

Malentendus fréquents (*)

- Système temps réel = système rapide et performant.
- La programmation temps réel = assembleur.
- Aucune science derrière le développement des systèmes temps réel, tout est une question de bidouillage.

(*) J. Stankovic, « Misconceptions about realtime computing »

Malentendus fréquents (*)

- L'augmentation de la vitesse des processeurs va résoudre les problèmes engendrés par le temps réel.

(*) J. Stankovic, « Misconceptions about realtime computing »

Problèmes rencontrés

- Allocation des ressources
- Architecture
- Méthode de développement

Problèmes rencontrés

- Allocation des ressources. Principal problème : *ordonnancement*.
  - Allocation des tranches de temps processeur.
  - But : assurer le respect des échéances de temps de manière prouvable.
  - Il faut construire une méthode d'ordonnancement.

Problèmes rencontrés

- Architecture :
  - Pour supporter un logiciel temps réel, l'architecture (hardware + OS) doit être *prédictible* : temps d'exécution des instructions, changement de contexte, accès mémoire, …
  - => pas de mémoire cache, de processeur superscalaire. Communications rapides et prédictibles (pas d'ethernet par exemple.)

Problèmes rencontrés

- Méthode de développement :
  - Spécification des besoins : *fonctionnelle* (ce que doit faire le système) et *non-fonctionnelle* (en combien de temps doit-il le faire) : spécifications conjointes et précises.
  - Conception et développement : langage de programmation, support de la communication et de la concurrence, constructions pour permettre de vérifier à la compilation le critère d'ordonnancabilité.