Institut
Mines-Télécom

Noyaux &
Noyaux temps réels

Thomas Robert
INF 341 - ETER

Rôle et services associés à un noyau

Pour créer la page « intercalaire », choisir la disposition « Page intercalaire » (2e choix)

Rappel :
Objectifs d'un système d'exploitation

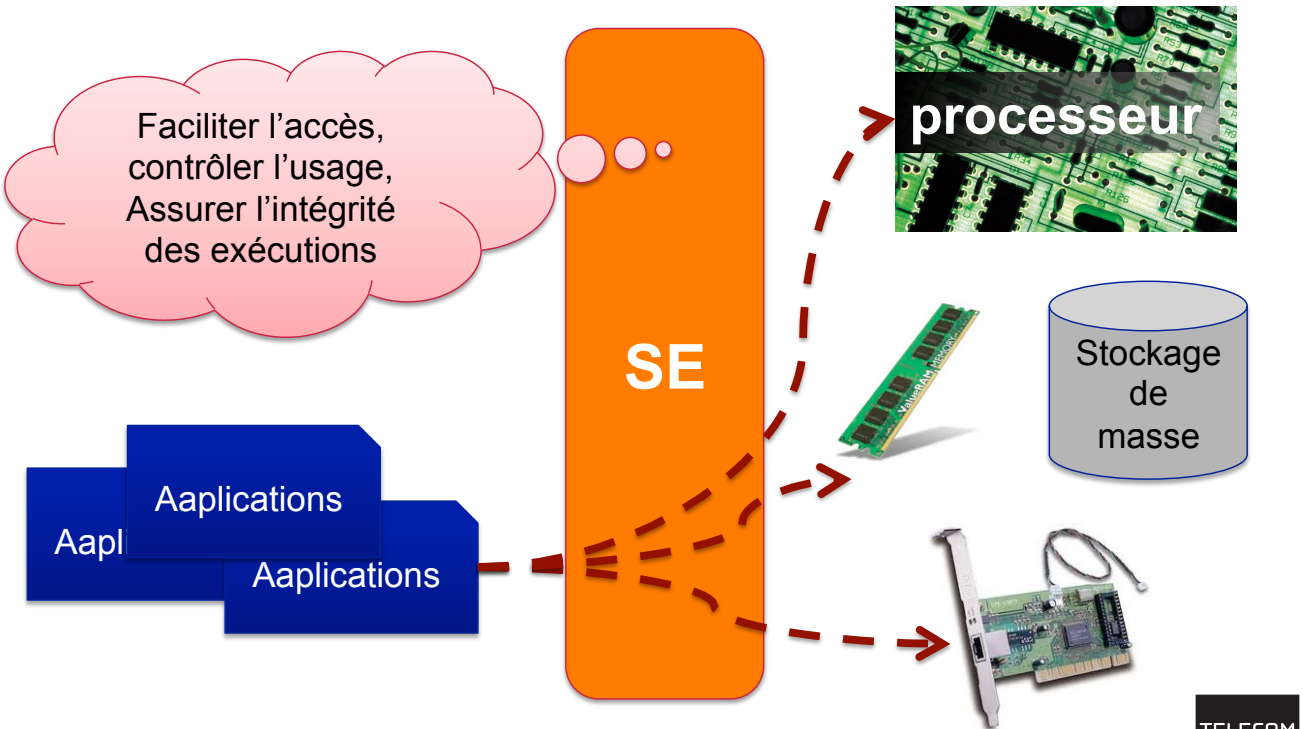
Gérer l'usage des ressources matérielles

- **Ressources de calcul**
- **Ressources de stockage**
- **Ressources d'interaction (périphériques)**
- **Ressources de communications (réseau/ipc)**

Sous des contraintes

- **D'intégrité des exécutions**
- **D'efficacité de l'allocation des ressources**
- **De contrôle des activités**

Rappel :
Objectifs d'un système d'exploitation (illustration)



Services Clés d'un système temps réel

- **Le contrôle d'exécution**
  - L'ordonnanceur
  - La gestion des exceptions, fautes et interruptions
- **La gestion mémoire, l'accès aux stockages de masse (le cas de la flash)**
  - Hiérarchie mémoire, allocation et placement
  - Pagination : contrôle d'accès
- **Les entrées/sorties et communication**
  - Les communication inter processus
  - Les entrées sorties lentes (polling vs interruptions)
  - Le drivers : performances / SdF
- **Support à la tolérance aux fautes**
  - Modes d'exécution / MMU
  - Virtualisation

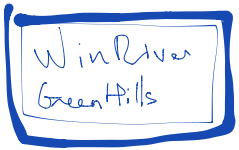
Quelques éléments d'architecture

Calculateur :

- **Unité d'exécution**
  - Registres : de travail (AX,BX…), de contrôle d'exécution (PC, SR,…), d'accès à la mémoire (SP, CS,SS…)
  - Pipeline d'exécution : latence = accès aux données, temps d'exécution variable
  - Gestion des interruptions
- **Adressage & mémoire**
  - Mémoire physique == { adresses }
  - Cache == mémoire rapide contenant une partie du contenu de la mémoire physique
  - MMU : composant dédié assurant le découpage {adresses} pour contrôle d'accès à la granularité d'une page

Contrôle d'exécution

Multi-tâches et contexte d'exécution



- **Fonction vs tâche**
  - Fonction : séquence d'opérations + procédure de transferts de paramètres (entrées), et résultat

  ⇒ Activité « réactive »
  - Tâche : séquence d'opérations + **événements d'activation + accès aux ressources** (optionnel : événement de terminaison asynchrone)
- **Contexte d'exécution**
  - État du processeur en rapport à l'exécution de la séquence d'opération
  - Structures de données permettant l'accès aux ressources (e.g. table des pages pour la mémoire).

Principe du changement de contexte

- **Exécuté par le système d'exploitation**
- **Logique en 4 étapes clés**
  1. Sauvegarde du registre PC en mémoire (sur la pile), via un registre auxiliaire, par registre dupliqués (banked registers)
  2. Sauvegarde complète du reste du contexte (e.g. registres configurant l'accès à son espace d'adressage) dans des structures de données dédiées (potentiellement relié à la notion de descripteur de tâche)
  3. Chargement du contexte de T2 depuis la mémoire (accès à l'espace d'adressage de T2, privilèges et restauration des registres de travail importants) + placement dans la sauvegarde de PC (cf 1) de la dernière valeur utilisée pour T2
  4. Sortie de l'appel système => restauration du PC)

Décomposition d'un ordonnanceur

- **Gestion des événements d'ordonnancement**
- **Cœur de la politique d'ordonnancement : « l'élection de la prochaine tâche »**
- **Changement de contexte**

Les événement d'ordonnancement

- **Les événements hérités du modèle de tâche :**
  - Activation
  - Terminaison
  - Synchronisation
  - Échéances
- **Les événements hérités de la politique d'ordonnancement**
  - Quantum
  - Détection d'erreur temporelle
  - …

On peut définir des règles de précédence qui permettent de ne pas bloquer l'exécution d'un thread. (T_1 doit s'exécuter n fois plus souvent que T_2).

Cas non préemptif (OSEK VDX)

- **Le SE réagit aux événements du modèle de tâche :**
  - Terminaison : élection d'une nouvelle tâche à exécuter
  - Synchronisation : élection d'une nouvelle tâche à exécuter
- **Le SE doit engendrer les événements suivants pour**
  - Échéances : terminaison des tâches en «dépassement » (si actives ou exécutées)
  - Activation : enregistrement des tâches dans la ready queue (pas de réélection => peut attendre la terminaison de la tâche en cours d'exécution)
- **La mise en œuvre de la politique repose uniquement sur les événements du modèle de tâche**

=> Implémentable par des appels explicites depuis le corps des tâches.

Cas préemptif dit tickless (RTAI)

- **Le SE réagit aux événements suivants pour :**
  - Terminaison : élection d'une tâche
  - Synchronisation : élection d'une tâche si suspendu ou si la politique préconise un ré-ordonnancement.
- **Le SE doit engendrer les événements suivants pour :**
  - Activation : enregistrement de la tâche et élection de la tâche à exécuter (préemption)
  - Échéance : terminaison de la tâche si elle est en cours d'exécution ou encore active=> élection d'une tâche
- **Génération des événements : utilisation d'un timer programmable pour calculer « le prochain réveil » cf TP ADA RT**

Election de la prochaine tâche.

1 point unique d'entrée (appelé par l'OS) (lors d'un événement généré, ou par les taches) + param pour identifier le contexte d'appel.

N point d'entrée par type d'evts déclencheur (séparation sync/async)

2. phases :

. Parcours des descripteurs de tache pour sélection de la tache
- parcours reposant sur des filtres.
- ?

Implémentation classique de l'élection de la prochaine tâche

- **1 point unique d'entrée (appelé par l'OS (lors d'un événement généré, ou par les tâches) + paramètres pour identifier le contexte d'appel**
- **N points d'entrée par type d'événement déclencheur (séparation activation/synchro)**
- **2 phases :**
  - parcours des descripteurs de tâches pour sélection de la tâche
    - Parcours reposant sur des files / listes
    - Filtrage en fonction de la priorité (attribut d''une tâche)
  - Mise à jour de l'attribut priorité si nécessaire en fonction du contexte d'appel (PCP / PIP ..)

Contrôle d'intégrité d'exécution

- **Les mécanismes temporel (gérés par le SE)**
  - Le gestionnaire de budget (détection exec time > WCET)
  - Le watchdog (vérification que le programme repasse bien par un point précis du code périodiquement)
- **Les exceptions du Pipeline d'exécution**
  - Faute : implémentation d'un masquage (e.g. défaut de page ou cache miss…)
  - Execption sur les opérandes : eg division by 0 implémentation forward recovery : execution d'un code visant à « signaler l'erreur » sans la corriger

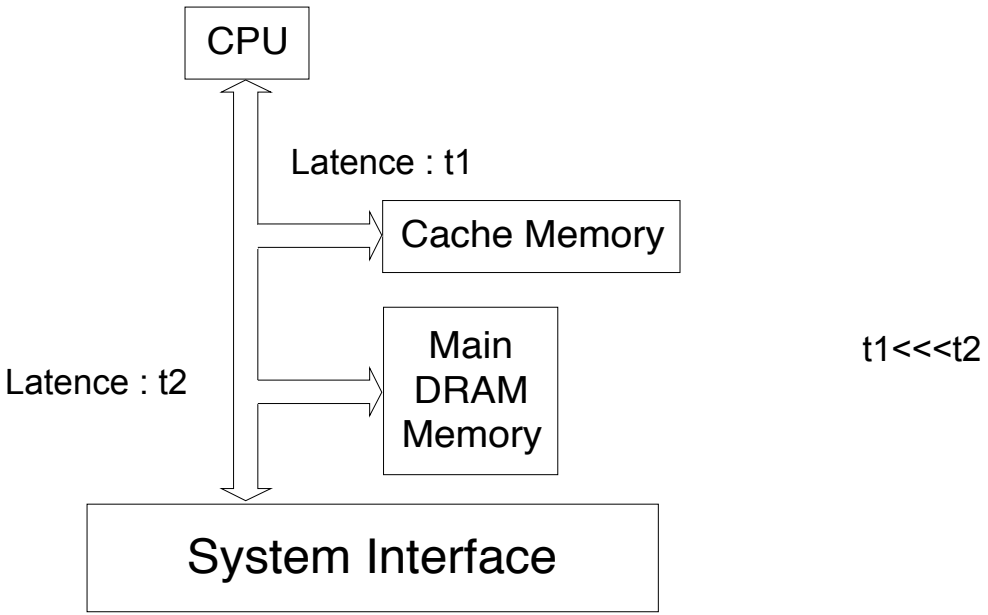
=> Dans le cas ou le problème n'est pas masqué le traitement doit être spécifié dans le SE
Mise en pratique dans OSEK via les hooks.

Cas préemptif dit à base de tick (OSEK VDX)

- **Le SE réagit aux événements suivants pour :**
  - Terminaison : élection d'une tâche
  - Synchronisation : élection d'une tâche si suspendu ou si la politique préconise un ré-ordonnancement.
- **Le SE doit engendrer les événements suivants pour :**
  - Activation : enregistrement de la tâche et élection de la tâche à exécuter (préemption)
  - Échéance : terminaison de la tâche si elle est en cours d'exécution ou encore active=> élection d'une tâche
  - Quantum : élection de la prochaine tâche
- **Génération des événements : utilisation d'un timer à période fixe (tp OSEK).**

Gestion de la mémoire et des stockages

Rappels sur la hiérarchie mémoire



Mécanismes usuellement déployés

- **Gestion de la mémoire virtuelle**
  - Traduction
- **Gestion du swap disque / mémoire physique de tout ou partie des données d'une tâche**
- **Gestion du ou des espace(s) d'adressage(s)**
- **Contrôle d'accès au contenu de la mémoire**
- **Édition de lien dynamique (rapport avec la gestion mémoire : la factorisation de code)**

Maitrise des latences d'accès

- **Les services en questions :**
  - Le swap des données applicatives entre disque et mémoire => si pagination toute page est présente en mémoire physique
  - L'édition de liens dynamique (référence calculée à la volée aux bibliothèques => risque de variabilité non maitrisée sur le temps d'exécution lors du chargement
  - La traduction logicielle des adresse virtuelles (page walk)
- **Les mécanismes du processeur en questions :**
  - Les caches et leurs politiques de remplacement
  - Le parcours HW de la table des pages

Le cas du « swap »

- **Principe de fonctionnement**
  - 1 table d'association @V / @P ou position sur stockage de masse
  - Occupation de la mémoire physique par un sous ensemble de la mémoire référençable
- **Temps d'accès à une @**
  - + Latence de traduction @V -> @P si échec

  => + Latence de swap
  - + Latence d'accès à la mémoire
- **Remarque sur l'isolation et les performances**
  - Si gestion globale du swap : Temps accès T1 non indépendant de l'activité de T2
  - Si gestion partitionnée du swap : # défaut de traduction plus important (résultat assez trivial).
- **Résolution du dilemme => désactivation du service par réservation statique de plage mémoire**

Factorisation de code et édition de lien dynamique

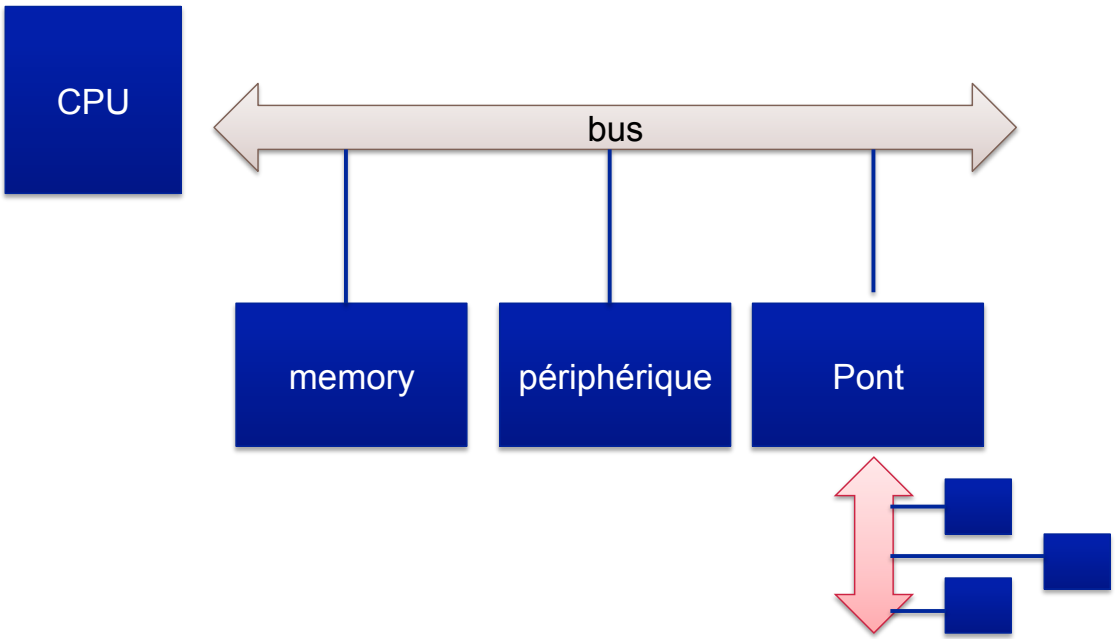
- **Le principe :**
  - Au chargement du code (à l'initialisation de son contexte d'exécution) ou lorsque l'on atteint une référence à la bibliothèque
    - Exécution d'un code générique pour
    - Rechercher la bibliothèque référencée pour
      - La chargée en mémoire si absente
      - Établir le lien entre chaque référence à son contenu et son placement en mémoire
  - A chaque accès : résolution de la référence à la bibliothèque partagée (cout temporel dépend de l'implémentation)
- **Vision dynamique de la factorisation de code non compatible avec la maitrise des temps d'exécution**

Gestion des espaces d'adressage

- **Principe pour instancier des espaces d'adressage:**
  - 1 structure de traduction @V /@P
  - 1 granularité permettant le partage de données / code
- **Remarques :**
  - Duplication des références à un même domaine d'adresses physiques
  - SI implémentation avec différent niveaux d'indirection => latence de traduction variable

Gestion des entrées / sorties

Architecture interne



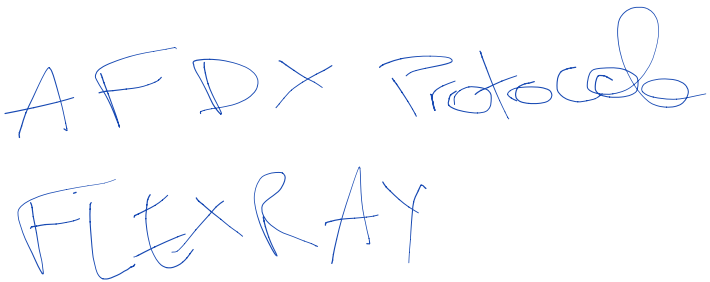
- **Que trouve-t-on sur une carte en plus du CPU**
  - Les blocs
    - PIC ou APIC
    - Bridges et autres co-processeurs
    - Timers, watchdogs, et horloges
    - Contrôleurs de bus (PCI, I2C, Série RS232)
    - Convertisseurs A/D ← analogic/Digital (Robotique)
    - PWM I/O (modulation par largeur de créneaux).
    - Contrôleur DMA
  - Les bus : largeur fréquence et synchronisation
- **Le processeur et les blocs dialoguent via ses registres.**
  - En MMIO : accès au registre par @
  - EN PMIO : accès via des instruction spécialisées

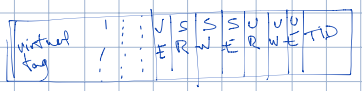
E/S modalité et support des interaction

- **Modalité d'interaction :**
  - synchrone
    - Accès direct à la mémoire interne du périphérique
    - Le processeur communique avec le bloc par lecture écriture dans sa mémoire interne (au bloc)
  - Asynchrone
    - Interruption permettant au bloc de se « signaler » au processeur => déroutement de son exécution pour traitement de l'interaction (cf plus haut)
    - Via des transferts de données en « background » via des contrôleurs mémoire particuliers e.g. DMA
- **Problématique propre à l'embarqué RT:**
  - Comment maitriser la latence d'accès à une E/S => protocoles de bus de terrain (cf CAN)
  - Comment empêcher les interruptions de polluer l''ordonnancement (cf classement des interrupts CAT1/ CAT2 dans osek).

Références

- **Considérations sur l'implémentation Trampoline de OSEK**
- **http://trampoline.rts-software.org/IMG/pdf/trampoline.pdf**





Praesium :

1 registre
- identification du contexte d'exécution (TID/PID)
- 1 registre pour le mode (U/S)