Un noyau multitâche temps réel: fournit les primitives minimums pour la gestion de processus en tenant compte des CT

22/11/2022 Noyau Temp réel:

Interface d'entrée = reçoit les requêtes

Gestionnaire d'objet ≈ gestion de la mémoire. Gère l'état des différents objets en maintenant leur cohérence en agissant sur des structures de données

L'ordonnanceur = chargé de gérer l'exécution des tâches en affectant le processeur de manière dynamique et réactive

Dispatcher (Pacceur) chargé d'activer la tâche active en restaurant son contexte d'exécution

Gestionnaire d'interruption: intercepte les interruptions, les acquitte et effectue le traitement des interruptions

Les objets d'un noyau:

La tâche: objet qui peut être actif dans l'application. C'est une séquence d'actions élémentaires.

Le descripteur de tâche (Task ou Process Control Block) contient un nom ou numéro, état, pointeur de programme, registres du CPU, info sur la mémoire (limite, page, segment), délais d'attente, durée d'utilisation processeur, priorité, pointeurs sur les files d'ordonnancement, Etats des E/S

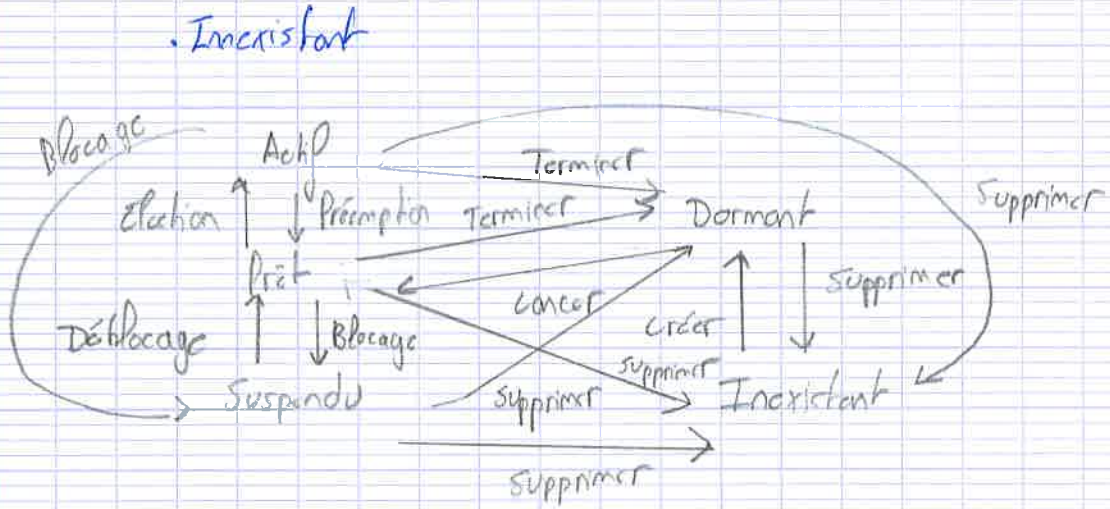
Objets liés au temps: Timer, Horloge Temps Réel

Objets courants: Lock ou Mutex (sémaphore moins courants) et Messages pour des choses complexes.

Etats d'une tâche:

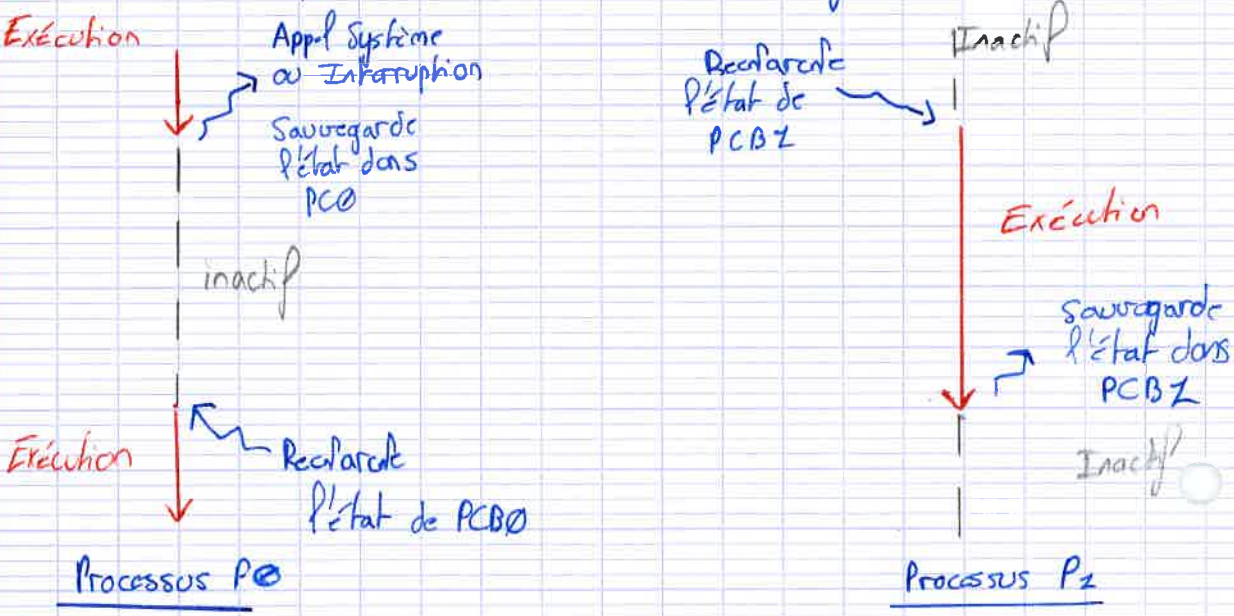
- Actif
- Prêt
- Suspendu
- Dormant

q. Machine à état pour les états d'une tâche



La commutation de contexte :

Contexte : Tout ce qui permet à une tâche de s'exécuter (liste des instructions, pointeur d'instruction, mémoire des variables et registres)



Remarque :
Le nombre de Sauvegardes et de Recharges de contextes doit être limités.

Critères de Performances :

Latence du Système : Délais global entre l'occurence d'un événement et la réaction attendue en sortie du système (ex : lancement d'une tâche). Elle prend en compte :
Le délais de scrutation (vérif d'interruption), liés au système d'exploitation (accès aux fichiers, chargement de contexte), de traitement, de transmission

de message (envoi sérialisé ie. bit à bit sur une carte réseau, demande de répétition par l'émetteur).
Rq: les protocoles type Ethernet, IP ne garantissent pas le TR la plupart du temps.

$L_{atence} = t_{cpu} + t_{wait}$

$t_{cpu} = \sum t_{exc} + \sum t_{spinlocks}$

t_{exc} : délais d'exécution ; $t_{spinlocks}$: scrutation

$t_{wait} = \sum t_{I/O} + \sum t_{sem} + \sum t_{int} + \sum t_{preempt}$

$t_{I/O}$: traitement des demandes d'E/S ; t_{sem} : exclusion mutuelle ; t_{int} : traitement des interruption ; $t_{preempt}$: commutation de contexte

spinlocks : désigne la vérification des primitives logiciel protégeant les ressources

sem : attente d'avoir accès à la ressource par le principe des sémaphores

Interruptions: Dans le cas de deux interruptions quasi-simultanées : ISR 1 et 2 sont les fonctions de traitement des interruptions lorsque l'OS les a remarqué
Le message d'interruption est enlevé du registre d'entrée et stocké ailleurs par le noyau.

Les ordonnancements Temps Réels mécanisme par lequel le noyau choisit la tâche à achever avec, dans le cas TR, 2 objectifs :

- en fonctionnement nominal, assurer le respect des CT de toutes les tâches
- en fonctionnement anormal, assurer l'exécution des tâches les plus critiques nécessaires à la sécurité du système

Avec en plus des objectifs de performance :

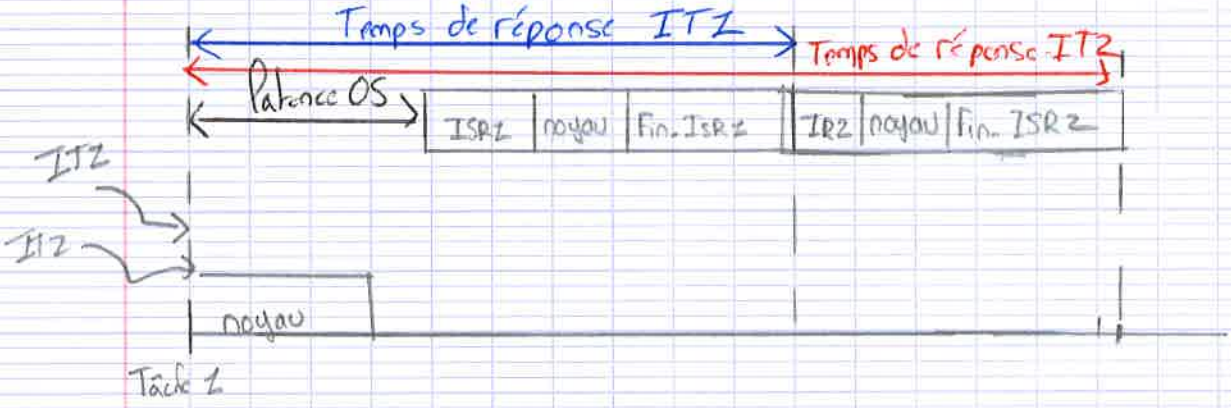
- minimiser le temps de réponse

9

- réduire la gigue de certaines tâches (variabilité de la latence)
- équilibrer la charge des sites (cas réparti) : exploiter le parallélisme

Propriétés des algorithmes :

- Séquence Valide : ordonnancement respectant les contraintes de toutes les tâches
- Ordonnançabilité : il existe au moins un algorithme capable de fournir une séquence valide
- Optimal : capable de fournir une séquence valide pour toute configuration de tâches ordonnançable



Catégories d'ordonnancement :

- Hors ligne : Liste préétablie => ordre d'exécution prédéfini
  Mise en œuvre simple

- En ligne : Non prédéfinie, avantage de pouvoir communiquer avec des procédés extérieurs non TR stricts

- Non préemptif : L'exécution d'un tâche ne peut pas être interrompue

- Préemptif : L'ordonnanceur peut réquisitionner le processeur en faveur d'une autre tâche

- Statique : Les priorités de tâches sont fixées à priori

• Dynamique : Les priorités des tâches dépendent de paramètres

Ordonnancement de tâches partageant des ressources

Le problème d'inversion de priorités :

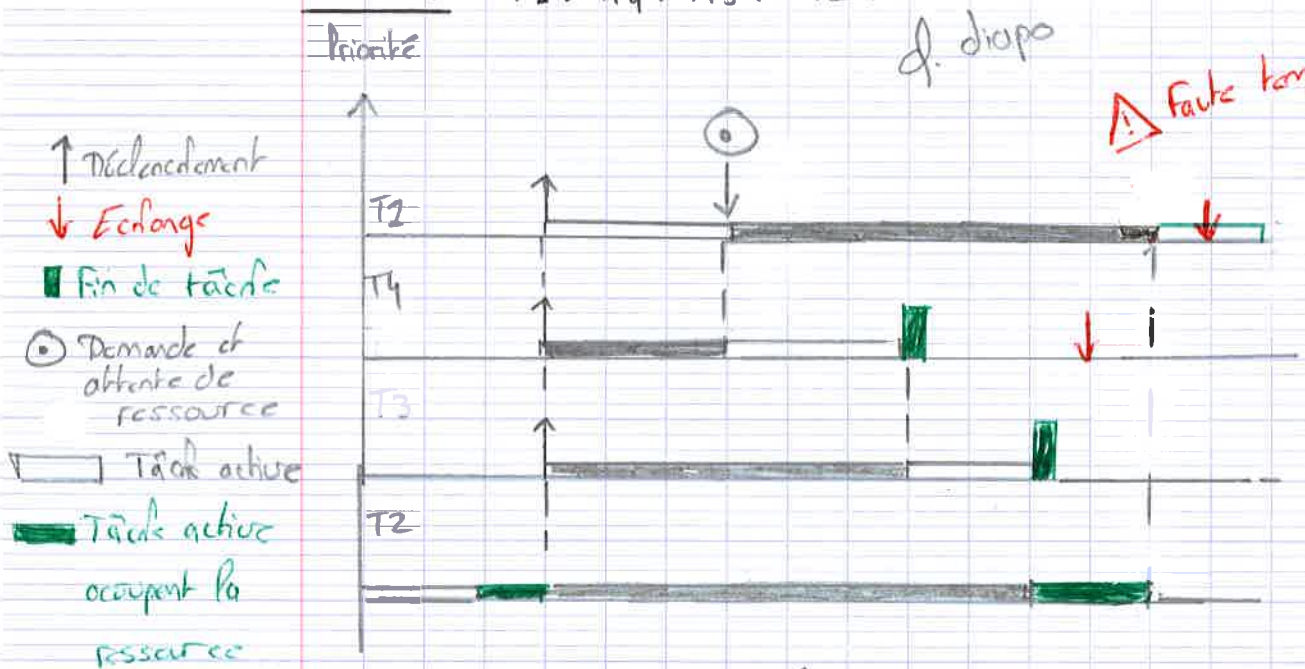
1. Une tâche T2 de priorité faible (20) demande et prend un sémaphore S
2. Une tâche T2 de priorité plus élevée (2) demande le sémaphore S
3. La tâche T2 est mise en attente sur S
4. T2 reprend son éxécution
5. Une tâche T3 de priorité intermédiaire (10) se réveille
6. T3 s'éxécute alors que T1 reste toujours en attente

→ Toute tâche de priorité plus élevée que T2 pourra s'éxécuter sans réveiller T2 => problématique pouvant bloquer le système.

Scénario : $P_{T2} > P_{T4} > P_{T3} > P_{T2}$:

Priorité

d. dispo



Exemple : Incident de Pathfinder en 1997

Watchdog : sous-système veillant sur le dépassement d'échéances et redémarre le système en cas d'incident

Dans le cas spatial : donc disponible pour des diagnostics et vérification sur terre.

L'héritage de priorité : Permet d'éviter le problème d'inversion de priorité => mécanisme de mutex à héritage. ⚠ complexité

1. Une tâche T2 de faible priorité (20) accède et réserve le mutex à héritage Mx.
2.

d exemple

Priority Ceiling Protocol : PCP (plafond de la priorité)

L'héritage de priorité :

↳ donne une borne max qui peut être inacceptable
↳ chaîne de blocage possible

=> intérêt du PCP : ressource partagée ⟶ une priorité plafond égale au Max (Prio) des tâches qui la partagent se voit attribuer

Une tâche Ti ne peut accéder à une ressource que si

{ elle est libre
et
Prio (Ti) > Max/plafond des ressources occupées

d exemple

⚠ Demande de R2 par T1

Immediate Priority Ceiling Protocol : IPCP :

PCP + Priorité dynamique de tâches

$P_{dynamique}(T_i) = Max[P(T_i); P(\text{ressources occupées par } T_i)]$

d. exemples

Caractéristiques temporelles d'une tâche T_i :

r_i : date d'activation — s_i : date de début d'exécution

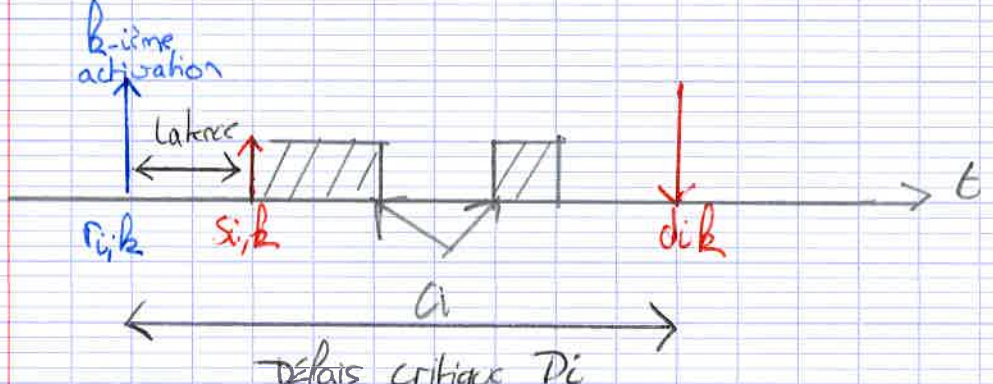
D_i : délais critique — C_i : durée d'exécution

T_i : période par tâche — d_i : échéance relative à la tâche

$d_i = r_i + D_i$

TR_i : temps de réponse de la tâche i

d. dapo



La laxité de Ti correspond à la marge de manoeuvre pour Ti à l'instant t

$L_i(t) = \text{Laxité}(t) = D_i(t) - C_i(t) = D_i + r_i - t - C_i(t)$

avec $C_i(t)$: durée d'exécution restante à l'instant t

$D_i(t)$: délais critique résiduel à l'instant t

La gigue temporelle d'une tâche est donnée par cf. diapo

$$\text{Gigue}_i = \max_{k} \left\{ \frac{|TR_{i,k} - TR_{i,k+1}|}{D_i} \right\}$$

Configuration de tâches : ensemble de n tâches partageant des ressources pour leur exécution. On considère :

- Début simultané ou échelonné

cf. diapo

.

Tâches non périodiques :

Tâches sporadiques : on ne leur connaît pas de périodes mais on connaît l'intervalle minimum entre deux requêtes successives (ex : réception de message sur un réseau). Leurs caractéristiques sont :

↳ C_{spor} = temps processeur nécessaire à son exécution

P_{spor}

Modèle de base :

- Nombre de tâches fixe
- Tâches périodiques et indépendantes (pas de partage de ressources, synchronisation, communication)
- Temps de commutation négligeable

cf. diapo