DATE S M T W T F S

e) les états des taches :

- Pour passer d'une tache T_1 à une tache T_2, l'ordonnanceur doit d'abort vérifier si la tache 2 est "prête" à etre executée.

- une tache n'est pas toujour prete à etre executée, par exemple si la Tache T_1 a fini son timing, et la tache T_2 est une acquisition de données qui recevait des valeurs éxterieurs (pendant execution de T_n) mais qui n'a pas fini de recevoir les données.

→ donc supposons que pendant execution de la tache T_2 le processeur va exploiter ces données.
Dans ce cas, la tache T_2 ne peut pas démarrer on dit qu'elle est "bloquée".

→ Lorsque une tache est prête à etre exploitée on dit qu'elle est exploitable ("runnable"). Ensuite lorsque l'ordonnanceur décide de lancer la tache 2, on dit qu'elle est exploitée : etat "running"

Donc, on déduit qu'une tache peut rester dans un état executable (runnable), qu'elle est donc prête à etre executée, mais elle ne s'éxecute pas jusqu'à ce que l'ordonnanceur décide d'executer cette tache (en fonction des critères par exemple).

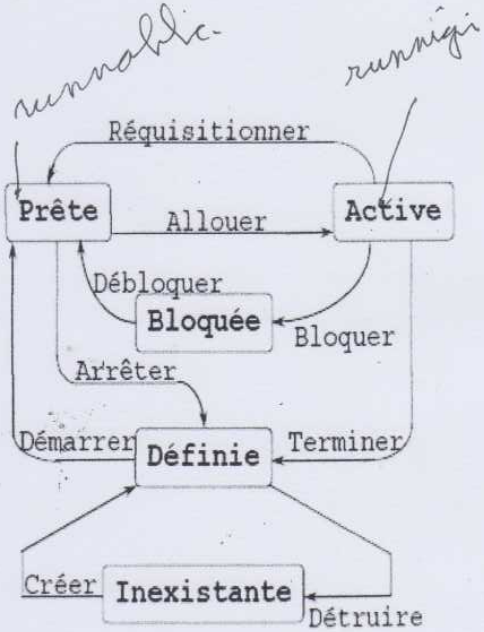
* Dans le schema suivant on va résumer les etats que peut prendre une tache dans un STR.

* Remarque:
comme pour l'ordonnancement, la préemption a besoin aussi de signal d'interruption pour qu'elle soit executée.

Etat d'une tâche

- Création d'une tâche => Nom + descripteur

- **Définie**
  - connue du superviseur
  - descripteur spécifié
- **Prête**
  - en attente du processeur
- **Active**
  - en cours d'exécution
- **Bloquée**
  - en attente d'une ressource



Revision: jusqu'à maintenant on a vu:

1 - definition du STR
2 - déroulement d'un ordonnancement
3 - ordonnancement préemptif et non préemptif
4 - Les Interruptions
5 - Les états des taches.

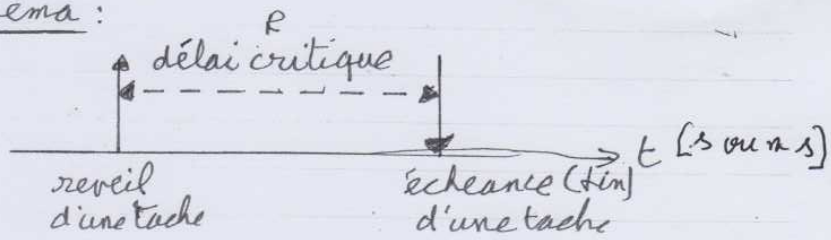
4) Description des Taches :

Taches periodiques - Taches apériodique

Introduction

- Chaque tache dans un STR passède un "Délai critique" R ; temp maximal pour s'executer depuis sa date de réveil. Le moment ou la tache doit s'arreter s'appel "Echeance". d_i ($i=0,1,\dots$)
- Si une tache depasse sont temps d'écheance on parle alors de "faute temporelle pour l'ordonnancement.

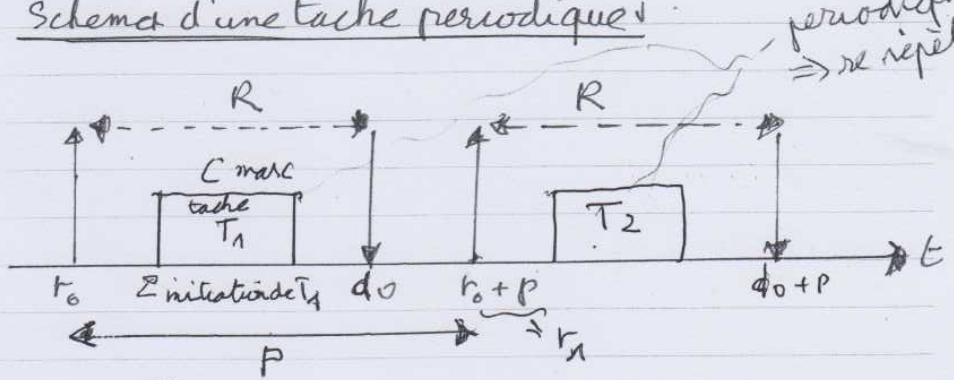
→ Schéma :



a) → Tache Périodique :

- Elles correspondent sur les mesures sur les procédés
- Elles se reveillent regulierment toute les periodes (en secondes). Ces taches sont les plus frequentes dans les STR : exemple de tache périodique : [gestion de bus].

Schema d'une tache périodique



on definit:

r_0 : date premier reveil $\boxed{r_1 = r_0 + P}$

C : temps d'execution d'une tache.

R : delai critique

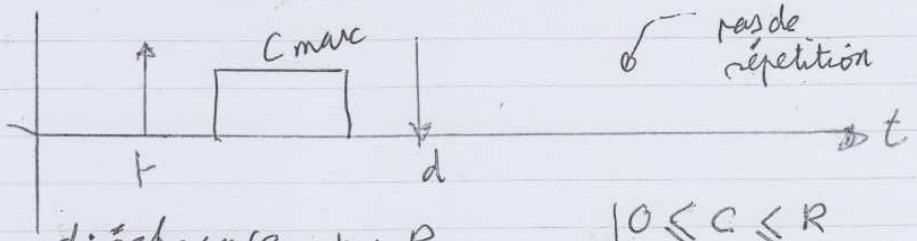
d_k : échéance : $\boxed{d_k = r_k + R}$ $k = 0, 1, 2, \ldots$

Nous avons : $\boxed{0 \leqslant C \leqslant R \leqslant P}$ → pour tache periodique

b) Tache aperiodique :

Elles correspondent aux événements,

Elles se reveillent de manière aléatoires.



d : échéance = $r + R$

$\boxed{0 \leqslant C \leqslant R}$ ↪ pas de P!

Exemple de Tache AP analyse

5) Algorithmes d'ordonnancement:

* Un algorithme d'ordonnancement est une méthode ou stratégie utilisée pour ordonnancer un processus réel.
* Le choix de l'algorithme d'ordonnancement s'appuye sur la connaissance de certaines caractéristiques du processus réel qu'on va commander par le STR, donc savoir si:
  1- processus périodique ou apériodique (processus avec des Taches periodique ap-)
  2- préemption ou non pour le processus
  3- système à priorité fixe ou à échéances.

→ Deux algorithmes classiques pour l'ordonnancement sont les plus connus:

a) RTM Rate Monotoring: (ne change pas lors de STR)

c'est un algorithme à priorité fixe (statique) qui donne la priorité d'ordonnancement à la tache qui a la frequence la plus élevée (la plus petite période (P)).

(th 0211.pdf) (TRP.pdf)

b) EDF (Earlest Deadline First):

* on définit la laxité L(t) : c'est le retard maximum d'une tache par rapport à son échéance (sa fin).
* Cet algorithme donne la priorité à la tache qui a la plus petite laxité.

(pas STR pour parler aux à des taches avec des retards)

* on revient à l'algorithme rate Monotrig et on définit :

Test d'ordonnançabilité :

C'est une condition suffisante pour tester si l'ordonnancement des taches est accepté

Soit : $\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n\left(2^{\frac{1}{n}} - 1\right)$

n : nombre de taches
P : la période des taches.
C_i : temp d'excéussion des taches.

Exemple :

Soit l'exemple suivant d'ordonnancement de trois taches $[T_1, T_2, T_3]$. chaque tache possède sont temps d'exécution $[C_i]$ et sa période $[P_i]$.

Q) Vérifier si l'ordonnancement par la méthode "Rate monotoring" est acceptable.

DATE

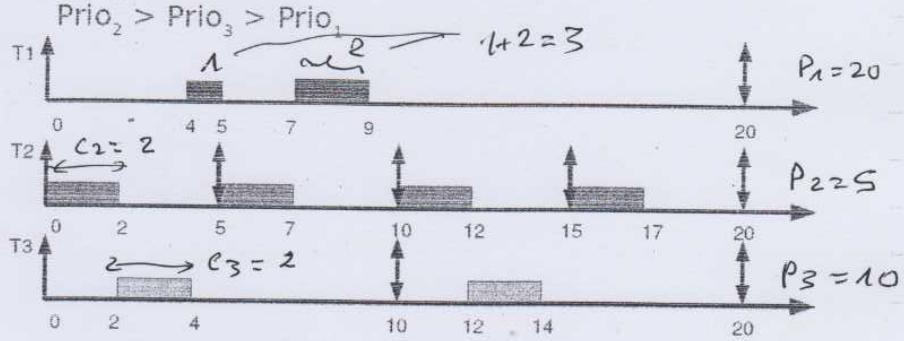
* les trois taches respectent leurs contraintes temporelles. [ne dépassent pas leurs délai grace à cet ordonnancement]

- exemple

T_1 (r_0=0, C=3, P=20), T_2 (r_0=0, C=2, P=5), T_3 (r_0=0, C=2, P=10)

$$\sum \frac{C_i}{P_i} = 3/20+2/5+2/10 = \boxed{0,75 \leq n(2^{1/n}-1)} = 0,77$$

$Prio_2 > Prio_3 > Prio_1$



* Dans cet schema de taches, la tache T_2 (P_2=5) est la tache la plus prioritaire (car son P_i le plus petit), et la tache T_1 (P_i = 20) est la tache la moins prioritaire.

* la methode RTM a été efficace pour cet exemple (test d'ordonnançabité ⇒ respect des contraintes temporelles), mais pour un autre exemple il n'y a pas de garantie que la méthode [RTM] soit efficace.