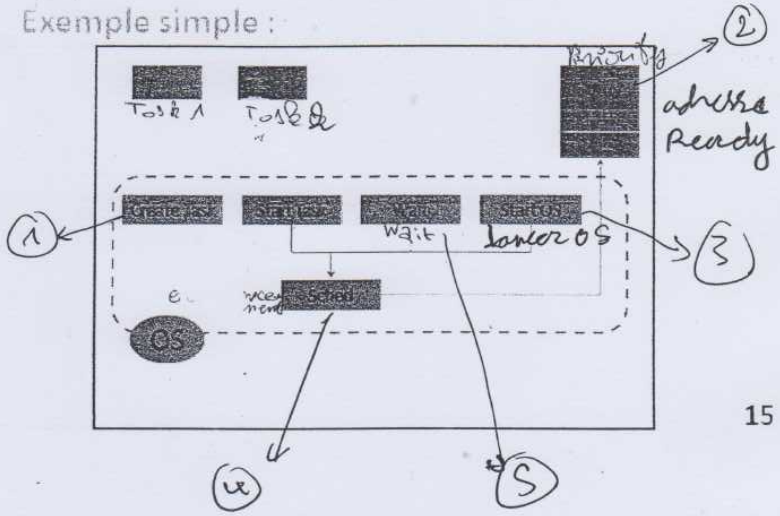
5- Exemple d'applications nécessitant un RTOS :

Exemple simple :



Systèmes Temps-Réel (STR)

5- Exemple d'applications nécessitant un RTOS :

Exemple simple :

dans cet exemple on retient les éléments suivants utilisé pour réaliser ce RTOS

-Task : la Tache

-Priorité de tache

-Interruption (d'une tache pour passer à une 2ème tache)

-Scheduler : le planificateur / ordonnanceur

DATE S M T W T F S

• Exemple Simple d'un OS qui execute un multitache

* Supposons on a deux taches : tache 1 et tache 2 ;
- et le but comment faire le code (programme) pour réaliser un multitache Tache 1, Tache 2.
* En faite le but est faire rouler les deux taches l'une après l'autre.

→ Pour realiser ce multitache, on va passer par les étapes suivantes ;

déclarer
(1) ← 1) Première chose c'est créer les taches ; c.ad programmer qu'il y a premiere tache qui s'appel tache 1, et une 2ème tache qui s'appel tache 2.

(2) créée
(2) ← 2) Créer une tache implique aussi de créer une table pour chaque tache ;
on va faire : la priorité
l'adresse
le Ready : execution
↳ tache 1 qui va se lancer ou bien waited (stoppée)

(4) executé
(3) ← 3) une fois cette portie faite, on va lancer l'OS : l'OS va appeler le service qui énumère (2)
(4) ← l'ordonnanceur ; qui va appeler les tables de tache 1 / il va chercher quelle est la priorité la plus haute, si elle est prête à se lancer ;
→ Donc il va chercher son adresse et il va la lancer

5) → On va imaginer que la tache 1 va se lancer et à un moment donné (après le timing connu d'avance) la tache 1 va appeler le service Wait (S)

- Le service wait va appeler le service ordonnanceur (Scheduler) qui va venir sur la table de la tache 1 et va mettre son etat à Wait
- etat Wait ⇒ tache 1 va stopper

※ Wait n'agit pas directement mais à travers l'ordonnanceur. ~~Si une tache appel le service wait~~

6) ※ L'ordonnanceur va chercher ensuite (dans la table) quel est la tache qui est prioritaire / dans notre cas si c'est la tache 2, il va lancer Ready pour la tache 2.

※ c'est comme ça que ça se passe et en temps réel.

Si on résume les étapes de cet ordonnancement
a) on crée les taches → déclarer [create T1]
b) on fait les tables : le déroulement de chaque tache et sa priorité
c) on lance l'OS ; il va juste chercher quelle est la tache la plus prioritaire.

(3) (tache 1)

(6) ⇒ si une tache appel le service [wait], elle va se stopper momentanement jusqu'à ce que une autre tache (tache 2) va starter d'elle même l'ordonnanceur pour switcher.

DATE . . . . . S M T W T F S

types de temps réel :

Il existe plusieurs types de (Systemes) temps réel :

1) Temps réel dure : [Hard Real time]

- Dans ce STR, le non respect des contraintes temporelles entraine la faute (défaut) au systeme de commande. par exemple trafic aerien.

2) Temps réel souple (Soft Real Time)

Dans ce Systeme, le respect des échéances (timing des taches et les taches elles mêmes) est imortante, mais s'il n'est pas (~~toujours~~) respecté, ça ne sera pas grave sur la commande du processus.

par exemple : Système d'acquisition de données pour affichage.

3) Temps Réel ferme, c'est soit ;

a - un TR souple : mais sans aucun retard dans les échéances.

ou

b - un TR dure : pour lequel quelques échéances peuvent etre occasionellement manquées

par exemple : projection vidéo.

DATE S M T W T F S

* Dans la réalité : un STR inclus généralement en même temps ces 3 types de contraintes temporelles (dure, souple, ferme)

* Temps concret : c'est l'horloge du Système informatique.

• Dans une application (ou Système) temps réel il faut pouvoir manipuler le temps concret (horloge).

* Donc le temps réel se fait sur la base du temps concret (H). il se fera soit :

a) en définissant le moment ou l'action (tâche) échéance) doit être commencée.

ou b) en définissant le moment ou l'action doit être finie.

* Pendant l'exécution d'un STR il peut être nécessaire de modifier ces paramètres (début d'action, fin d'action), comme il peut être nécessaire de pouvoir préciser l'action à prendre en cas de faute temporelle