Module Master M1

Systèmes temps réel et Informatique Industrielle
Chapitre XII : Révision sur les systèmes temps réel

Présenté par : Prof. Kholladi Mohamed-Khireddine
Département d'Informatique
Facultés des Sciences Exactes
Université Echahid Hamma Lakhdar d'El Oued
Tél. 0770314924
Email. kholladi@univ-eloued.dz et kholladi@yahoo.fr
Site Web. www.univ-eloued.dz
http://kholladi.doomby.com/ et http://kholladi.e-monsite.com/

XII - Révision sur les systèmes temps réel

XII.1 – Introduction

À la demande de certaines étudiantes du Master M1 de 2017 à 2020, j'ai préparé une petite séance de révision des cours sous la forme d'exercices avec des réponses pour leur donner une certaine idée sur les questions lors des interrogations écrites ou des contrôles de connaissances.

XII.2 – Exercice 1

1. Etablir le schéma des quatre états de transition des tâches (ou des processus) aux sens de la norme POSIX ainsi que les flèches de transition possible ?
2. Dans ce contexte, qu'est-ce qu'un processeur ?
3. Et par la même occasion, qu'est-ce qu'une tâche (ou un processus) ?
4. Donner les familles de tâches possibles et expliquer le comportement.
5. Quels sont les paramètres de tâche i (dans le cadre de la formalisation d'un système temps réel)? Mais avant quelle est la contrainte de la tâche i suivant le diagramme d'exécution?

Solution de l'exercice 1

1. Le schéma de transition des quatre états des tâches (ou des processus) est comme sur la figure XII.1.

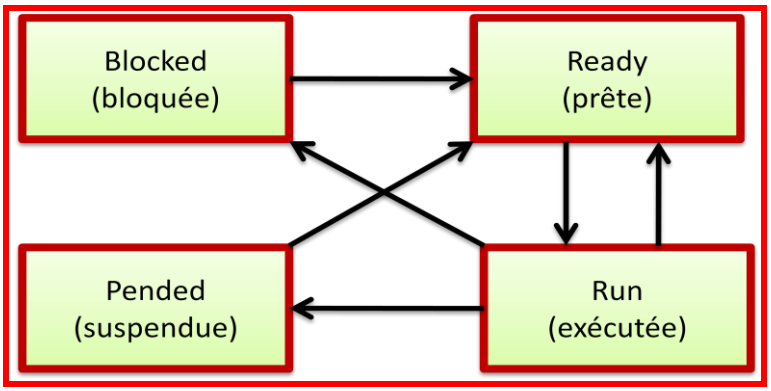

Figure XII.1 – Le schéma de transition des quatre états des tâches

2. Dans ce contexte, qu'est-ce qu'un processeur?

Dans ce cas, un processeur est une unique ressource partageable dans un système temps réel. C'est un exécutif temps réel.

3. Par la même occasion, qu'est-ce qu'une tâche (ou un processus) ?

Une tâche (ou un processus) est une suite d'instructions, plus les données, plus un contexte d'exécution (état).

4. Donner les familles de tâches possibles et expliquer le comportement.

Les familles de tâches sont :

- Tâches dépendantes ou indépendantes.
- Tâches importantes ou non.
- Tâches urgentes ou non.
- Tâches répétitives : activations successives (tâches périodiques ou sporadiques), ce qui entraine la gestion des fonctions critiques.
- Tâches non répétitives/apériodiques : dans ce cas, une seule activation, ce qui implique la gestion des fonctions non critiques.

5. Quels sont les paramètres de tâche i (dans le cadre de la formalisation d'un système temps réel)? Mais avant quelle est la contrainte de la tâche i suivant le diagramme d'exécution?

La tâche i doit terminer son traitement avant le $t_i + D_i$ comme sur la figue XII.2.

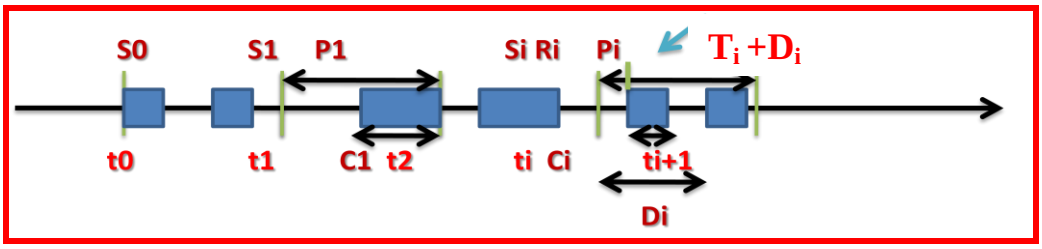

Figure XII.2 – Les paramètres de la tâche i

Les paramètres définissant la tâche i comme sur la fifure XII.3 :

- Arrivée de la tâche dans le système : S_i.
- Pire temps d'exécution d'une activation : C_i (capacité).
- Période d'activation : P_i.
- Délai critique : D_i (relatif à P_i si tâche périodique, à S_i si tâche apériodique).
- Date d'exécution au plus tôt : R_i.

Le tableau suivant et la figure XII.3 résume les paramètres d'une tâche.

Variables ou formules	**Signification**
r	Date de réveil - moment du déclenchement de la première requête d'exécution.
C	Durée d'exécution maximale (capacité).
D	Délai critique : délai maximum acceptable pour son exécution.
P	Période (si tâche périodique).
d = r+D	Échéance (si tâche à contraintes strictes).
$r_k = r_0 + k*P$	Tâche périodique : si D = P, tâche à échéance sur requête.

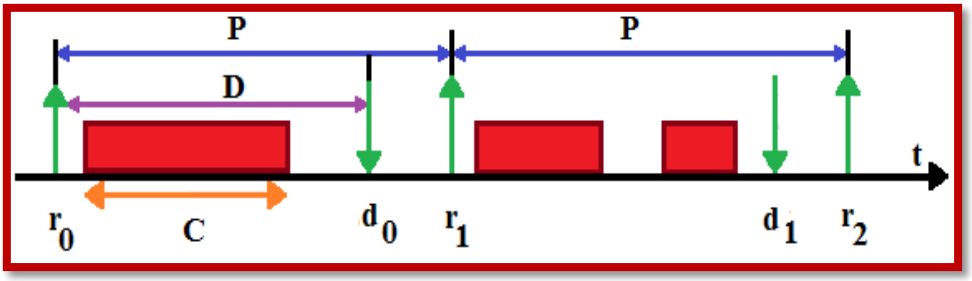

Figure XII.3 – Les paramètres de

XII.3 – Exercice 2

1. Qu'est-ce que l'algorithme EDF ? et expliquer brièvement le principe de l'algorithme EDF.
2. Quelles cinq caractéristiques essentielles de l'algorithme EDF ? et détailler brièvement chacune des cinq caractéristiques.

Solution de l'exercice 2

1. Qu'est-ce que l'algorithme EDF ? et expliquer brièvement le principe de l'algorithme EDF.

L'algorithme EDF c'est un algorithme d'ordonnancement des tâches à priorités dynamiques. Il signifie Earliest Deadline First (EDF).

2. Quelles cinq caractéristiques essentielles de l'algorithme EDF ? et détailler brièvement chacune des cinq caractéristiques.

Les cinq caractéristiques essentielles de l'algorithme EDF sont :

1) C'est un algorithme d'ordonnancement des tâches à priorité dynamique. Il est mieux adapté que l'algorithme d'ordonnancement des tâches à priorité fixe (ou statique) surtout pour les applications dynamiques.
2) Il supporte les tâches périodiques et les tâches apériodiques.
3) C'est un algorithme optimal, car il utilise jusqu'à 100% de la ressource du processeur.
4) Sa mise en œuvre est difficile dans un système d'exploitation en général et dans un système d'exploitation temps réel en particulier.
5) C'est un algorithme instable en cas de surcharge du processeur. Il est moins déterministe que l'algorithme d'ordonnancement des tâches à priorité fixe (ou statique).

XII.4 – Exercice 3

1. Quels sont les objectifs de l'ordonnancement d'un système temps réel ?
2. Donner la taxinomie (la classification) des éléments de l'ordonnancement d'un système temps réel.

Réponse de l'exercice 3

1. Quels sont les objectifs de l'ordonnancement d'un système temps réel ?

Les objectifs de l'ordonnancement temps réel sont de prendre en compte les besoins d'urgence et d'importance des applications temps réel.

2. Donner la taxinomie (la classification) des éléments de l'ordonnancement d'un système temps réel.

   Les éléments de taxinomie ou de classification de l'ordonnancement temps réel sont :

   1) Algorithmes hors ligne/en ligne : moment où sont effectués les choix d'allocation.
   2) Priorités statiques/dynamiques : les priorités changent elles ?
   3) Algorithmes statiques/dynamiques.
   4) Algorithmes préemptifs ou non : tâches interruptibles par d'autres ?
   5) non préemptif =
      a) Exclusion mutuelle aisée ou facile des ressources.
      b) Surcoût de l'ordonnanceur moins élevé.
      c) Efficacité moindre.

XII.5 – Exercice 4

1. Quelles sont les quatre propriétés principales recherchées d'un algorithme d'ordonnancement d'un système temps réel ?
2. Quelles sont les cinq catégories de composants logiciels??
   - Détaillez votre réponse avec des exemples.
   - Dans un tableau à trois colonnes (composants, explication, exemples).

Solution de l'exercice 4

1. Quelles sont les quatre propriétés principales recherchées d'un algorithme d'ordonnancement d'un système temps réel ?

   Les quatre propriétés principales recherchées d'un algorithme d'ordonnancement temps réel sont :

   1) Faisabilité/ordonnançabilité : est-il possible d'exhiber un test de faisabilité ? Condition permettant de décider hors ligne du respect des contraintes des tâches. Exemple : pire temps de réponse des tâches.

2) Optimalité : critère de comparaison des algorithmes (un algorithme est dit optimal s'il est capable de trouver un ordonnancement pour tout ensemble faisable de tâches).
3) Complexité : les tests de faisabilité sont ils polynômiaux ? exponentiels ? passage à l'échelle ?
4) Facilité de mise en œuvre : l'ordonnanceur est-il facile à implanter dans un système d'exploitation ?

2 Quelles sont les cinq catégories de composants logiciels ?

- Détaillez votre réponse avec des exemples.
- Dans un tableau à trois colonnes (composants, explication, exemples).

Les cinq catégories de composants logiciels sont :

Composants	Explications	Exemples
Thread (fil ou flot)	Fil d'exécution (flot de contrôle qui exécute un programme) tâche Ada/VxWorks, thread POSIX/Java, etc.	thread receiver end receiver; thread implementation receiver.impl end receiver.impl; thread analyser thread implementation analyser.impl
Data (donnée)	Structure de données implantée dans le langage cible	struct C, class C++/Java, record Ada, etc.
Process (Processus)	Modélise un espace mémoire (protection mémoire). Il doit contenir au moins un thread.	process processing end processing; process implementation processing.others
Subprogramm sous programme)	Modélise un programme exécutable séquentiellement. Il est associé à un code source. fonction C, méthode Java, sous-programme Ada, etc.	subcomponents receive : thread receiver.impl; analyse : thread analyser.impl; end processing.others;
Thread group (groupe de fils)	Modélise la notion de hiérarchie entre threads	

XII.6 – Exercice 5

1. Quelles les quatre catégories de composants matériels pour les systèmes temps réel ? Dans un tableau à trois colonnes (composants, explication, exemples).
2. Quel est le processus de vérification et de validation d'un système temps réel ?

Solution de l'exercice 5

1. Quelles les quatre catégories de composants matériels pour les systèmes temps réel? Dans un tableau à trois colonnes (composants, explication, exemples).

- Les quatre catégories de composants matériels pour les systèmes temps réel sont :

Composants	Explications	Exemples
Processor (processeur)	Abstraction du logiciel/matériel en charge de l'ordonnancement des threads. Un processeur peut comporter plusieurs virtuels processeurs.	Microprocesseur
Memory (mémoire)	Modélise toute entité de stockage physique de données	Disque dur, mémoire vive, etc.
Device (dispositif)	Composant qui interagit avec l'environnement. On ignore sa structure interne (thread, data, etc.).	Capteur, actionneur device antenne end antenne; processor leon; end leon;
Bus	Entité permettant l'échange de donnée/contrôle entre device, memory ou processor	Réseau de communication, bus, etc.

2. Quel est le processus de vérification et de validation d'un système temps réel?

Le processus de vérification et de validation d'un système temps réel est le suivant:

1) On définit l'architecture matérielle et l'environnement d'exécution : la capacité mémoire, le processeur et sa puissance de calcul, le système d'exploitation (et donc l'algorithme d'ordonnancement des tâches).
2) On réalise le code fonctionnel (ex: fonction C d'un thread POSIX).
3) On conçoit l'architecture logicielle: comment associer le code fonctionnel et le matériel. Conduit à modéliser les tâches du système ainsi que leurs contraintes et caractéristiques temporelles.
4) On valide la faisabilité de l'architecture logicielle sur l'architecture matérielle, c'est-à-dire, entre autre la faisabilité du jeu de tâches.
5) Eventuellement, on revient à 1, 2 ou 3. Cycle de conception / vérification.

XII.7 – Exercice 6

1. Donner le schéma d'exécution d'un modèle simplifié de tâches périodiques synchrones à échéance sur requête selon les trois étapes.
2. Quelles sont les deux caractéristiques? Détailler votre réponse en utilisant la formulation du schéma de l'exercice 1.
3. Quelles sont les trois fonctions principales?

Solution de l'exercice 6

1. Donner le schéma d'exécution d'un modèle simplifié de tâches périodiques synchrones à échéance sur requête selon les trois étapes.

La figure XII.4 illustre le schéma d'exécution d'un modèle simplifié de tâches périodiques synchrones à échéance sur requête selon les trois étapes.

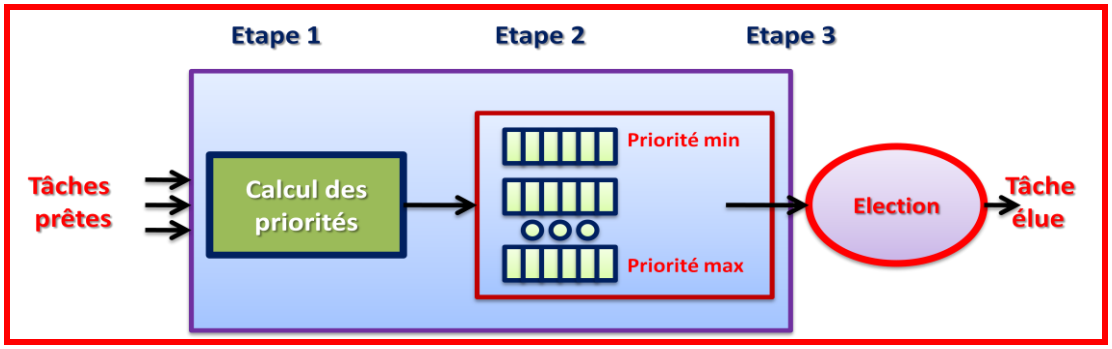

Figure XII.4 – Le schéma d'exécution d'un modèle de tâches périodiques

2. Quelles sont les deux caractéristiques? Détailler votre réponse en utilisant la formulation du schéma de l'exercice 1.

Les deux caractéristiques sont :

1. Tâches périodiques.
2. Tâches indépendantes.
   - avec ∀i : Si = 0 ce qui implique un instant critique (pire cas).
   - avec ∀i : Pi = Di ce qui implique des tâches à échéance sur requête.

3. Quelles sont les trois fonctions principales?

Les trois fonctions principales sont :

1. Calcul de priorité :

2. En ligne ou hors ligne.
3. Période : Rate Monotonic (RM).
4. Echéance : Deadline Monotonic (DM), Earliest Deadline First (EDF).
5. Laxité : Least Laxity First (LLF). (laxité à l'instant t: Li (t) = Di (t)− reliquat de Ci à exécuter).
6. Etc.

XII.8 – Exercice 7

1. Quels sont les deux algorithmes classiques d'ordonnancement des processus ou des tâches?
2. Quelles sont les caractéristiques de l'ordonnancement des processus (ou des tâches) à priorité fixe (ou statique)?
3. Quel est le fonctionnement de l'ordonnancement des processus (ou des tâches) à priorité fixe (ou statique)?
4. Comment s'effectue l'affectation des priorités selon Rate Monotonic?
5. Comment faire cohabiter des tâches apériodiques dans un système ordonnancé par priorité fixe (ou statique) avec Rate Monotonic? Détailler votre réponse.

Solution de l'exercice 7

1. Quels sont les deux algorithmes classiques d'ordonnancement des processus ou des tâches?

   Les deux algorithmes classiques d'ordonnancement des processus (ou des tâches) sont :

   1) Algorithme à priorité fixe, avec affectation des priorités selon Rate Monotonic (RM, RMS, RMA).
   2) Algorithme à priorité dynamique, Earliest Deadline First (EDF).

2. Quelles sont les caractéristiques de l'ordonnancement des processus (ou des tâches) à priorité fixe (ou statique)?

Les caractéristiques de l'ordonnancement des processus (ou des tâches) à priorité fixe (ou statique) sont :

1) Priorité fixe, implique une analyse hors ligne pour des applications statiques et critiques.

2) Complexité faible et mise en œuvre facile dans un système d'exploitation.

3. Quel est le fonctionnement de l'ordonnancement des processus (ou des tâches) à priorité fixe (ou statique)?

Le fonctionnement de l'ordonnancement des processus (ou des tâches) à priorité fixe (ou statique) est le suivant :

1) Affectation hors ligne des priorités.
2) Election de la tâche de plus forte priorité.

4. Comment s'effectue l'affectation des priorités selon Rate Monotonic?

L'affectation des priorités selon Rate Monotonic s'effectue selon les conditions suivantes :

1) Algorithme optimal dans la classe des algorithmes à priorité fixe.

2) Tâches périodiques uniquement.

3) Priorité implique l'inverse de la période.

4) Selon les cas préemptif ou non.

5. Comment faire cohabiter des tâches apériodiques dans un système ordonnancé par priorité fixe (ou statique) avec Rate Monotonic? Détailler votre réponse.

Pour faire cohabiter des tâches apériodiques dans un système ordonnancé par priorité fixe (ou statique) avec Rate Monotonic, il faut savoir que :

1) Les tâches apériodiques ne sont pas urgentes, ce qui implique que les priorités plus faibles que les tâches périodiques.
2) Les tâches apériodiques sont urgentes, ce qui implique l'utilisation de serveur de tâches apériodiques.

XII.9 – Exercice 8

1. Dans un système de temps réel souple, dans quels cas le non-respect de l'échéance est acceptable?

2. Dans un système de temps réel strict, Dans quels cas le non-respect de l'échéance ne peut survenir?

3. Est-ce que les problèmes d'ordonnancement de recherche opération (RO) sont différents des problèmes d'ordonnancement temps réel (TR)? Pourquoi?
4. Citer trois exemples d'ordonnancement des tâches périodiques.
5. Citer un exemple d'ordonnancement des tâches apériodiques.
6. Par quoi s'effectue la synchronisation des tâches? Citer deux exemples.

Solution de l'exercice 8

1. Dans un système de temps réel souple, dans quels cas le non-respect de l'échéance est acceptable?

Dans un système de temps réel souple, le non-respect de l'échéance est acceptable dans les cas suivants:

1) Dans un certain pourcentage
2) Un certain nombre de fois
3) Avec une certaine fréquence
4) Peut aboutir à un traitement dégradé

2. Dans un système de temps réel strict, dans quels cas le non-respect de l'échéance ne peut survenir?

Dans un système de temps réel strict, le non-respect de l'échéance ne peut survenir dans les cas suivants :

1) Déterminisme maximal des opérations
2) WCET : Worst Case Execution Time
3) Réduction des points non-déterministes
4) Pré-allocation des ressources
5) Surdimensionnement du système

3. Est-ce que les problèmes d'ordonnancement de recherche opération sont différents des problèmes d'ordonnancement temps réel? Pourquoi?

Les problèmes d'ordonnancement de recherche opérationnelle (RO) sont différents des problèmes d'ordonnancement de temps réel (TR).

Le but étant pour la RO de minimiser (hors ligne) le temps de réponse et pour le TR de satisfaire (en ligne) des échéances.

4. Citer trois exemples d'ordonnancement des tâches périodiques.

   Les trois exemples d'ordonnancement des tâches périodiques sont :

   1) Ordonnancement à base de table
   2) Ordonnancement préemptif à priorité fixe
   3) Ordonnancement préemptif à priorité dynamique

5. Citer un exemple d'ordonnancement des tâches apériodiques.

   Un exemple d'ordonnancement des tâches apériodiques est :

   - Serveur sporadique

6. Par quoi peut s'effectuer la synchronisation des tâches? Citer deux exemples.

   La synchronisation des tâches peut s'effectuer par

   1) Héritage de priorité
   2) Priorité plafonnée

XII.10 – Exercice 9

1. Donner les trois conditions pour qu'une tâche (ou un processus) libère le processeur.
2. Donner le graphe du modèle d'état de transition du système temps réel (OSEK/VDK) en nommant les nœuds et les arcs de transition (en anglais).
3. Etablir le tableau à deux colonnes (état et description) des quatre états d'une tâche (ou d'un processus).
4. Etablir un tableau à quatre colonnes (transition, état précédent, état suivant, description) du modèle d'état de transition de tâche d'un système temps réel (OSEK/VDK).
5. Quand est-ce qu'il est possible d'avoir une fin d'une tâche (ou d'un processus)? Quelle est la conséquence ?
6. Qu'est-ce qu'on utilise pour résoudre le problème typique des mécanismes communs de synchronisation ? Quel est le problème ? Que signifie-t-il dans ce cas ?

Solution exercice 9

1. Donner les trois conditions pour qu'une tâche (ou un processus) libère le processeur.

   Les trois conditions pour qu'une tâche (ou un processus) libère le processeur sont :

1) Si elle finit;
2) Si le système d'exploitation change la tâche par une tâche de priorité plus élevée;
3) Ou si une interruption survient qui amène le processeur à exécuter une routine de gestion d'interruption (ISR).

2. Donner le graphe du modèle d'état de transition du système temps réel (OSEK/VDK) en nommant les nœuds et les arcs de transition (en anglais).

Le graphe du modèle d'état de transition du système temps réel (OSEK/VDK) en nommant les nœuds et les arcs de transition est le suivant comme sur la figure XII.5.

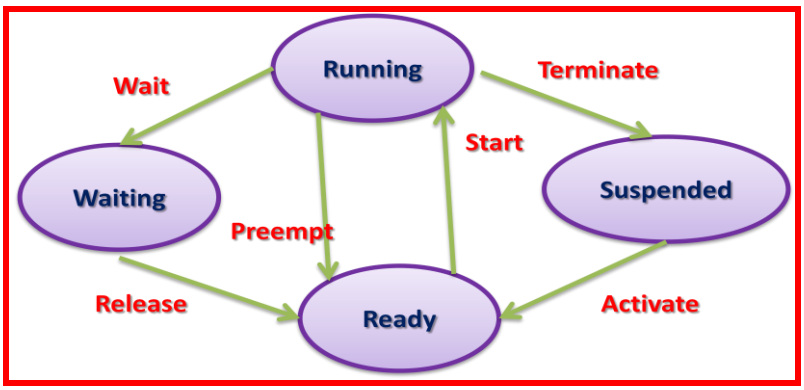

Figure XII.5 – Le graphe du modèle d'état de transition

3. Etablir le tableau à deux colonnes (état et description) des quatre états d'une tâche (ou d'un processus).

Le tableau à deux colonnes (état et description) des quatre états d'une tâche (ou d'un processus) est le suivant :

État s	**Descriptions**
Running	Dans l'état running, le CPU est assigné à la tâche, afin que ses instructions soient exécutées. Seule une tâche peut être dans cet état à un instant donné, alors que tous les autres états peuvent être adoptés simultanément par plusieurs tâches.
Ready	Tous les pré-requis pour une transition vers l'état running sont vérifiés, et la tâche attend seulement que le processeur lui soit alloué. L'ordonnanceur décide quelle tâche dans l'état ready sera exécutée ensuite.
Waiting	Une tâche ne peut continuer son exécution car elle doit attendre (wait) au moins un événement.

Suspended	Dans l'état suspended la tâche est inactive et peut être activée.

4. Etablir un tableau à quatre colonnes (transition, état précédent, état suivant, description) du modèle d'état de transition de tâche d'un système temps réel (OSEK/VDK).

   Réponse dans le tableau suivant :

Transition	**État précédent**	**État suivant**	**Description**
Activate	Suspended	Ready	Une nouvelle tâche entre dans l'état ready à la suite d'un service système. Le système d'exploitation OSEK assure l'exécution de la tâche à partir de sa première instruction.
Start	Ready	Running	Une tâche ready sélectionnée par l'ordonnanceur est exécutée.
Wait	Running	Waiting	La transition dans l'état waiting est causée par un des services du système. Pour pouvoir continuer l'opération, la tâche dans l'état waiting requiert un événement.
Release	Waiting	Ready	Au moins un événement est survenu, sur lequel la tâche attendait (waiting).
Preempt	Running	Ready	L'ordonnanceur décide de démarrer une nouvelle tâche. La tâche running est mise dans l'état ready.
Terminate	Running	Suspended	La tâche provoque sa transition dans l'état suspended par un service système.

5. Quand est-ce qu'il est possible d'avoir une fin d'une tâche (ou d'un processus)? Quelle est la conséquence?

La fin d'une tâche (transition terminate) n'est possible que lorsque la tâche se finit elle-même (self-termination, auto-désactivation).

Cette restriction réduit la complexité du système d'exploitation.

6. Qu’est-ce qu’on utilise pour résoudre le problème typique des mécanismes communs de synchronisation? Quel est le problème? Que signifie-t-il dans ce cas?

Pour résoudre le problème typique des mécanismes communs de synchronisation, on fait à l'usage des sémaphores.

Le problème est le problème d'inter blocage.

Dans ce cas, l'inter blocage signifie l'impossibilité d'exécution d'une tâche due à une attente infinie sur des ressources mutuellement bloquées.