ECOLE CENTRALE D'ELECTRONIQUE
GROUPE ECE

J. FRANCOMME

Eléments de correction

Systèmes multitâches et Temps réel

Contrôle avril 2006

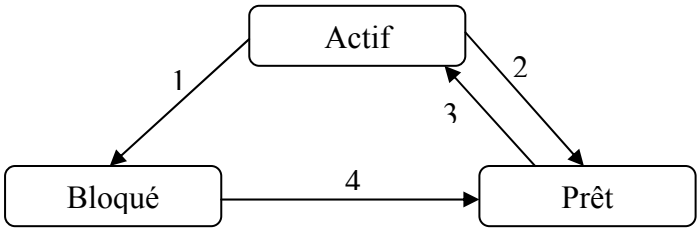
Durée : 2 heures

Tout document autorisé – Machine à calculer autorisée.
Vos réponses devront être les plus concises possible (évitez les longues dissertations)
Durée suggérée / barème indicatif (sur 20)
Repérez bien le numéro de chacune des questions sur votre copie.
La qualité de la présentation sera prise en compte dans le barème de notation.
N'oubliez pas de joindre les documents réponses à votre copie.

Cette correction du contrôle amène en même temps que les réponses des éléments permettant de comprendre la démarche. Il n'est pas demandé à l'étudiant de rédiger tout ceci. 2 heures ne suffiraient sûrement pas !
Bonne lecture. J. FRANCOMME

A. Système Multitâche et Temps Réel - Définitions (5 mn / 1 point)

a) Indiquez le nom de la transition se faisant entre l'état actif et l'état éligible pour une tâche.



Un processus peut se trouver dans différents états, suivant qu'il dispose de tout ou partie des ressources dont il a besoin pour s'exécuter. On distingue la ressource processeur de l'ensemble des autres ressources.

- *La transition de l'état actif vers l'état prêt (on dit aussi éligible) se nomme la préemption dans le cas d'un ordonnanceur préemptif : Un processus actif peut perdre le processeur, et repasser dans l'état prêt lorsque le système désire allouer le processeur à un autre processus.*
- *La transition de l'état actif à l'état prêt peut aussi s'appeler la « coopération » dans le cas d'un ordonnanceur non préemptif ; en effet le « scheduler » en pouvant arrêter la tâche, c'est la tâche elle-même qui doit décider de relâcher le processeur pour qu'une autre tâche puisse effectuer un peu de travail.*

b) Indiquez la différence entre une attente active et une attente passive pour une tâche. Illustrez chacun des deux cas avec un exemple simple de votre choix.

Lors d'une attente active, une tâche doit nécessairement disposer du processeur afin de sortir de cette situation. Cette tâche consomme donc du temps CPU alors qu'elle ne peut pas continuer son activité faute d'un évènement particulier.

Exemple : Une tâche A scrute une variable (un drapeau) afin de savoir si une tâche B est déjà arrivée à un point de « Rendez-vous ». Elle lit la variable tant que la valeur recherchée n'est pas positionnée.

Lors d'une attente passive, une tâche se retrouve bloquée alors qu'elle est en attente d'un évènement qui n'est pas encore arrivé. Cette tâche ne dispose donc plus du processeur. On ne gaspille plus de temps CPU pour l'attente. Les autres tâches actives peuvent profiter de ce temps pour avancer leur travail. C'est le scheduler qui décide du moment où la tâche devra être débloquée ; il est informé de l'arrivée de l'évènement et positionne alors la tâche à l'état éligible (déblocage).

Exemple : Une tâche A utilise un sémaphore afin de savoir si une ressource critique à un seul accès est disponible ; si celui-ci ne l'est pas, la tâche est bloquée par le mécanisme du noyau. Cette tâche sera débloquée dès lors que la tâche utilisatrice de la ressource critique aura relâché le sémaphore qui a bloqué la tâche A.

B. Ordonnancement – Ordonnancement statique et ordonnancement dynamique

B.1 Ordonnancement statique (5mn / 1 point) : On considère l'ordonnancement basé sur RM

Expliquez ce que l'on entend, lorsqu'on dit que la condition d'ordonnancement de Liu et Layland (Equ. 1) est une condition suffisante mais pas nécessaire.

$$U = \sum_{i=1}^{N} \left(\frac{C_i}{T_i} \right) \leq N(2^{\frac{1}{N}} - 1) \quad \textbf{(Equ. 1)}$$

avec :

N	$N(2^{\frac{1}{N}} - 1)$
3	0,78

Ordonnancement statique : on dit que la condition d'ordonnancement de Liu et Leyland est une condition suffisante mais pas nécessaire dans la mesure où la validation de cette condition indique l'ordonnançabilité statique inconditionnelle du système de tâches.

Elle n'est pas nécessaire dans la mesure où la condition non validée n'indique pas que l'ordonnançabilité statique du système de tâches est impossible. Il faudra dans ce cas utiliser un critère différent et valider plus finement la faisabilité de l'ordonnancement de ce système de tâches.

B.2 Ordonnancement statique (45 mn / 6 points) : Soient trois tâches périodiques τ_1, τ_2, τ_3 à échéances sur requêtes définies par les paramètres suivants :

Date de réveil de chacune des tâches : $r_1 = r_2 = r_3 = 0$

τ_1 : Période $T_1 = 29$, Temps d'exécution $C_1 = 7$

τ_2 : Période $T_2 = 5$, Temps d'exécution $C_2 = 1$

τ_3 : Période $T_3 = 10$, Temps d'exécution $C_3 = 2$

a) Indiquez ce que l'on entend lorsque l'on dit que les tâches sont à échéances sur requêtes.

On dit qu'une tâche est à « échéance sur requête » lorsque son échéance est égale à sa période : $D_i = T_i$.

b) Calculez le taux d'utilisation du processeur pour un ordonnancement RM pour ce système de tâches. Concluez sur l'ordonnançabilité du système de tâches.

On utilise la condition ci-dessus de Liu & Leyland qui permet de comparer le taux d'utilisation du processeur à une borne supérieure qui indique l'ordonnançabilité inconditionnelle du système de tâches.

Avec N=3

$$U = \sum_{i=1}^{3} \left(\frac{C_i}{T_i} \right) = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} \leq 3(2^{\frac{1}{3}} - 1)$$

A.N. :

$$U = \frac{7}{29} + \frac{1}{5} + \frac{2}{10} \leq 3(2^{\frac{1}{3}} - 1) \quad \Rightarrow \quad 0{,}641 \leq 0{,}779$$

La condition est donc bien vérifiée ; le système de tâches est inconditionnellement ordonnançable en RM.

c) Sur quel intervalle de temps minimum (période d'étude) doit-on vérifier l'ordonnançabilité d'un système de tâche ?

L'intervalle de temps minimum s'appelle l'hyper-période ; elle est déterminée par le PPCM $\{T_i\}$.
Dans notre exemple, PPCM {29, 5, 10}=290.

d) Déterminez le nombre d'unité de temps libre sur la période d'étude.

Sur l'ensemble de l'hyper-période, le processeur est utilisé à ≈ 64% (il faudra prendre la valeur exacte pour le calcul). Il faut simplement appliquer une règle de trois pour avoir la valeur recherchée.

Processeur utilisé à 100% sur une hyper-période correspond à 290 slots de temps.

Pour une utilisation de 64% : ? Mais comme on cherche le nombre d'unités de temps libre et non pas occupées, il faudra prendre le complément : c'est-à-dire (1-0,64).

Il vient donc : $(1-U)\times 290 = 104$ *unités de temps libre ! Ca fait pas mal de temps libre !!!!!!*

e) Confirmez les points précédents en dessinant sur les 40 premières unités de temps la séquence générée par RM, pour un ordonnanceur **non-préemptif** puis un ordonnanceur **préemptif** ***(cf. document réponse Figure 1 et Figure 2, page 8).***

Ordonnancement non préemptif : on remarque sur la figure 1, que la tâche 1 qui s'est approprié le processeur et désire terminer son travail avant de le rendre à ses copines, le garde trop longtemps et ne permet ainsi pas à la tâche 2 de pouvoir respecter son échéance.

Ce système de tâches ne fonctionne donc pas avec un ordonnanceur non préemptif (figure 1).

Le système de tâches fonctionne très bien avec un ordonnanceur préemptif. Toutes les tâches respectent leurs échéances. On note toutefois qu'il faut aller jusqu'au bout de l'hyper-période afin de prouver l'ordonnançabilité sous forme graphique (figure 2). Mais comme nous avons vérifié cela avec le critère de Liu & Leyland, on peut s'en passer.

Un ordonnancement non préemptif est généralement moins efficace qu'un ordonnancement préemptif.

On modifie la tâche τ_1 par période $T_1 = 30$, Temps d'exécution $C_1 = 6$ et la tâche τ_2 par $C_2 = 3$.

f) Calculez le nouveau taux d'utilisation du processeur (toujours dans le cas RM) pour ce système de tâches. Dessinez de nouveau la séquence d'ordonnancement générée par RM sur sa période d'étude en mode préemptif ***(cf. document réponse Figure 3, page 9)***. Que constatez-vous ?

On utilise la condition ci-dessus de Liu & Leyland :

Avec N=3

$$U = \sum_{i=1}^{3} \left(\frac{C_i}{T_i} \right) = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} \le 3(2^{\frac{1}{3}} - 1)$$

A.N. :

$$U = \frac{6}{30} + \frac{3}{5} + \frac{2}{10} \le 3(2^{\frac{1}{3}} - 1) \quad \Rightarrow \quad 1 \le 0{,}779 \; !!!$$

La condition suffisante n'est pas validée ; cela ne veut pas dire que le système ne soit pas ordonnançable. Cela veut simplement dire que le taux d'utilisation du processeur est trop important pour garantir de façon fiable l'ordonnançabilité du système de tâches. Il faudra donc prendre son courage à deux mains et vérifier pour chaque tâche distinctement, qu'elle respecte bien son échéance. L'outil à utiliser pour évaluer cette faisabilité est le test de Lehoczky cité ci-dessous.

g) Confirmez ce dernier résultat en calculant uniquement le temps de réponse de la tâche la moins prioritaire à l'aide du test de terminaison (Equ. 2).

$$\forall i, 1 \le i \le N, \min_{0 \le t \le D_i} \sum_{j=1}^{i} \frac{C_j}{t} \left\lceil \frac{t}{T_j} \right\rceil \le 1 \qquad \textbf{(Équ. 2)}$$

Avec $W_i(t) = \sum_{j=1}^{i} C_j \left\lceil \frac{t}{T_j} \right\rceil$ et $t_0 = \sum_{j=1}^{i} C_j$

Dans le cas d'un ordonnancement RM, la tâche la moins prioritaire est la tâche τ_3.
On évaluera donc la date de fin de cette tâche.
Sachant qu'elle est la moins prioritaire, et que l'on a rangé les tâches par ordre de priorité, on applique la condition nécessaire et suffisante avec comme temps de départ la valeur $t_0 = C_1 + C_2 + C_3 = 6 + 3 + 2 = 11$

$$i = 3, \min_{0 \le t \le D_i} \sum_{j=1}^{3} \frac{C_j}{t} \left\lceil \frac{t}{T_j} \right\rceil \le 1$$

priorité (τ_2)> priorité (τ_2)> priorité (τ_1)

$$\sum_{j=1}^{3} \frac{C_j}{t_0} \left\lceil \frac{t_0}{T_j} \right\rceil = \frac{C_1}{t_0} \left\lceil \frac{t_0}{T_1} \right\rceil + \frac{C_2}{t_0} \left\lceil \frac{t_0}{T_2} \right\rceil + \frac{C_3}{t_0} \left\lceil \frac{t_0}{T_3} \right\rceil = \frac{6}{11} \left\lceil \frac{11}{30} \right\rceil + \frac{3}{11} \left\lceil \frac{11}{5} \right\rceil + \frac{2}{11} \left\lceil \frac{11}{10} \right\rceil$$

$$= \frac{6}{11} \times 1 + \frac{3}{11} \times 3 + \frac{2}{11} \times 2 = 1{,}727 \ge 1, \text{ donc pas valable !}$$

On réitère jusqu'à trouver la borne recherchée avec un nouveau temps :

$$t_1 = \sum_{j=1}^{3} C_j \left\lceil \frac{t_0}{T_j} \right\rceil = C_1 \left\lceil \frac{t_0}{T_1} \right\rceil + C_2 \left\lceil \frac{t_0}{T_2} \right\rceil + C_3 \left\lceil \frac{t_0}{T_3} \right\rceil = 6 \times \left\lceil \frac{11}{30} \right\rceil + 3 \times \left\lceil \frac{11}{5} \right\rceil + 2 \times \left\lceil \frac{11}{10} \right\rceil$$

$$= 6 \times 1 + 3 \times 3 + 2 \times 2 = 19 < D_1, \text{ on peut donc continuer !}$$

$$\sum_{j=1}^{3} \frac{C_j}{t_1} \left\lceil \frac{t_1}{T_j} \right\rceil = \frac{C_1}{t_1} \left\lceil \frac{t_1}{T_1} \right\rceil + \frac{C_2}{t_1} \left\lceil \frac{t_1}{T_2} \right\rceil + \frac{C_3}{t_1} \left\lceil \frac{t_1}{T_3} \right\rceil = \frac{6}{19} \left\lceil \frac{19}{30} \right\rceil + \frac{3}{19} \left\lceil \frac{19}{5} \right\rceil + \frac{2}{19} \left\lceil \frac{19}{10} \right\rceil$$

$$= \frac{6}{19} \times 1 + \frac{3}{19} \times 4 + \frac{2}{19} \times 2 = 1{,}158 \geq 1, \text{ donc toujours pas valable!}$$

On réitère jusqu'à trouver la borne recherchée avec un nouveau temps :

$$t_2 = \sum_{j=1}^{3} C_j \left\lceil \frac{t_1}{T_j} \right\rceil = C_1 \left\lceil \frac{t_1}{T_1} \right\rceil + C_2 \left\lceil \frac{t_1}{T_2} \right\rceil + C_3 \left\lceil \frac{t_1}{T_3} \right\rceil = 6 \times \left\lceil \frac{19}{30} \right\rceil + 3 \times \left\lceil \frac{19}{5} \right\rceil + 2 \times \left\lceil \frac{19}{10} \right\rceil$$

$$= 6 \times 1 + 3 \times 4 + 2 \times 2 = 22 < D_1, \text{ on peut donc continuer !}$$

$$\sum_{j=1}^{3} \frac{C_j}{t_2} \left\lceil \frac{t_2}{T_j} \right\rceil = \frac{C_1}{t_2} \left\lceil \frac{t_2}{T_1} \right\rceil + \frac{C_2}{t_2} \left\lceil \frac{t_2}{T_2} \right\rceil + \frac{C_3}{t_2} \left\lceil \frac{t_2}{T_3} \right\rceil = \frac{6}{22} \left\lceil \frac{22}{30} \right\rceil + \frac{3}{22} \left\lceil \frac{22}{5} \right\rceil + \frac{2}{22} \left\lceil \frac{22}{10} \right\rceil$$

$$= \frac{6}{22} \times 1 + \frac{3}{22} \times 5 + \frac{2}{22} \times 3 = 1{,}227 \geq 1, \text{ donc toujours pas valable!}$$

On réitère jusqu'à trouver la borne recherchée avec un nouveau temps :

$$t_3 = \sum_{j=1}^{3} C_j \left\lceil \frac{t_2}{T_j} \right\rceil = C_1 \left\lceil \frac{t_2}{T_1} \right\rceil + C_2 \left\lceil \frac{t_2}{T_2} \right\rceil + C_3 \left\lceil \frac{t_2}{T_3} \right\rceil = 6 \times \left\lceil \frac{22}{30} \right\rceil + 3 \times \left\lceil \frac{22}{5} \right\rceil + 2 \times \left\lceil \frac{22}{10} \right\rceil$$

$$= 6 \times 1 + 3 \times 5 + 2 \times 3 = 27 < D_1, \text{ on peut donc continuer !}$$

$$\sum_{j=1}^{3} \frac{C_j}{t_3} \left\lceil \frac{t_3}{T_j} \right\rceil = \frac{C_1}{t_3} \left\lceil \frac{t_3}{T_1} \right\rceil + \frac{C_2}{t_3} \left\lceil \frac{t_3}{T_2} \right\rceil + \frac{C_3}{t_3} \left\lceil \frac{t_3}{T_3} \right\rceil = \frac{6}{27} \left\lceil \frac{27}{30} \right\rceil + \frac{3}{27} \left\lceil \frac{27}{5} \right\rceil + \frac{2}{27} \left\lceil \frac{27}{10} \right\rceil$$

$$= \frac{6}{27} \times 1 + \frac{3}{27} \times 6 + \frac{2}{27} \times 3 = 1{,}111 \geq 1, \text{ donc toujours pas valable!}$$

On réitère jusqu'à trouver la borne recherchée avec un nouveau temps :

$$t_4 = \sum_{j=1}^{3} C_j \left\lceil \frac{t_3}{T_j} \right\rceil = C_1 \left\lceil \frac{t_3}{T_1} \right\rceil + C_2 \left\lceil \frac{t_3}{T_2} \right\rceil + C_3 \left\lceil \frac{t_3}{T_3} \right\rceil = 6 \times \left\lceil \frac{27}{30} \right\rceil + 3 \times \left\lceil \frac{27}{5} \right\rceil + 2 \times \left\lceil \frac{27}{10} \right\rceil$$

$$= 6 \times 1 + 3 \times 6 + 2 \times 3 = 30 = D_1, \text{ on peut donc continuer !}$$

$$\sum_{j=1}^{3} \frac{C_j}{t_4} \left\lceil \frac{t_4}{T_j} \right\rceil = \frac{C_1}{t_4} \left\lceil \frac{t_4}{T_1} \right\rceil + \frac{C_2}{t_4} \left\lceil \frac{t_4}{T_2} \right\rceil + \frac{C_3}{t_4} \left\lceil \frac{t_4}{T_3} \right\rceil = \frac{6}{30} \left\lceil \frac{30}{30} \right\rceil + \frac{3}{30} \left\lceil \frac{30}{5} \right\rceil + \frac{2}{30} \left\lceil \frac{30}{10} \right\rceil$$

$$= \frac{6}{30} \times 1 + \frac{3}{30} \times 6 + \frac{2}{30} \times 3 = 1 \leq 1, \text{ donc VALABLE!}$$

On a une solution, mais est-ce vraiment le minimum ? On réitère jusqu'à trouver la borne recherchée avec un nouveau temps :

$$t_5 = \sum_{j=1}^{3} C_j \left\lceil \frac{t_4}{T_j} \right\rceil = C_1 \left\lceil \frac{t_4}{T_1} \right\rceil + C_2 \left\lceil \frac{t_4}{T_2} \right\rceil + C_3 \left\lceil \frac{t_4}{T_3} \right\rceil = 6 \times \left\lceil \frac{30}{30} \right\rceil + 3 \times \left\lceil \frac{30}{5} \right\rceil + 2 \times \left\lceil \frac{30}{10} \right\rceil$$

$$= 6 \times 1 + 3 \times 6 + 2 \times 3 = 30 = D_1, \text{ on peut donc continuer !}$$

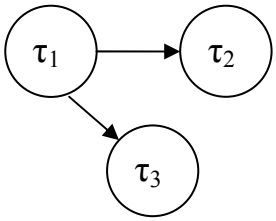
Comme on avait le même temps avec l'itération précédente, on peut sentir que le minimum est atteint.

$$\sum_{j=1}^{3} \frac{C_j}{t_4} \left\lceil \frac{t_4}{T_j} \right\rceil = \frac{C_1}{t_4} \left\lceil \frac{t_4}{T_1} \right\rceil + \frac{C_2}{t_4} \left\lceil \frac{t_4}{T_2} \right\rceil + \frac{C_3}{t_4} \left\lceil \frac{t_4}{T_3} \right\rceil = \frac{6}{30} \left\lceil \frac{30}{30} \right\rceil + \frac{3}{30} \left\lceil \frac{30}{5} \right\rceil + \frac{2}{30} \left\lceil \frac{30}{10} \right\rceil$$

$$= \frac{6}{30} \times 1 + \frac{3}{30} \times 6 + \frac{2}{30} \times 3 = 1, \text{ donc VALABLE!}$$

La tâche τ1 respecte son échéance et se termine à t = 30. Ce que l'on vérifie sur le la séquence illustrée sur la figure 3.

B.3 Ordonnancement de tâches dépendantes (5 mn / 2 points) : On considère maintenant arbitrairement que les 3 tâches précédentes sont dépendantes. Le graphe de précédence est indiqué ci-dessous.



τ_1 (r_1=0,C_1=7)
τ_2 (r_2=0,C_2=1)
τ_3 (r_3=0,C_3=2)
T_1=T_2=T_3=30

En quoi consiste la transformation de Chetto/BLazewicz pour le cas EDF rappelée ci-dessous ?
Remarque : il n'est pas demandé d'effectuer les calculs

$$r_i^* = Max\{r_i, Max\{r_j^* + C_j\}\},\ T_j \rightarrow T_i$$

$$D_i^* = Min\left\{D_i, Min\left\{D_j^* - C_j\right\}\right\},\ T_i \rightarrow T_j$$

La transformation de Chetto/Blazewicz consiste à transformer un système de tâches dépendantes en un système de tâches indépendantes sur lequel on peut appliquer tout ce que l'on a vu précédemment. Cette méthode propose de déterminer une nouvelle :

- *date de réveil pour chacune des tâches du systèmes permettant de garantir par exemple que la tâche τ_2 se réveillera au moins après le réveil de la tâche précédente (en l'occurrence τ_1) augmenté de sa capacité (pour le cas EDF) pour être sûr que la tâche τ_1 soit complètement terminée.*
- *échéance qui garantira qu'en respectant sa propre échéance, la tâche τ_1 permet à la tâche qui lui succède (en l'occurrence τ_2) de respecter sa propre échéance.*

C. Ordonnancement – Exercice de modélisation (50 mn / 7 points)

Remarque *: Il est important de lire toutes les questions de cet exercice avant de commencer.*

Contexte : On imagine qu'une secrétaire soit affectée dans un service scolaire ; on vous demande d'organiser sa journée de travail qui commence à 13h et se termine à 19h.

La secrétaire doit réaliser un certain nombre de tâches dans sa journée :

- Elle doit trier et distribuer le courrier. Le courrier arrive à 13h et à 16h. Le courrier de 13h doit être distribué avant 16h et celui de 16h doit être distribué avant 19h. Cette tâche lui demande 30 minutes à chaque fois.
- Elle a pour charge la réalisation de l'emploi du temps ; cela lui demande 2,5h de travail.
- Elle doit gérer l'accueil des étudiants et répondre, tant que possible, à leur demande administrative.
- Son contrat de travail lui octroie des pauses tout au long de la journée. Pour chaque période de 2h de travail, elle peut stopper le travail pendant 15 minutes. Le service disposant d'une badgeuse, elle peut fractionner ses pauses de 15 minutes comme bon lui semble du moment qu'elle ne dépasse pas 15 minutes toutes les 2 heures.

Pour commencer, on ne tient pas compte du travail que nécessite l'accueil des étudiants. On considère que la secrétaire est capable d'interrompre la réalisation d'une tâche si une autre plus prioritaire intervient.

a) La secrétaire (assimilée au processeur) peut-elle effectuer sa charge de travail tout en respectant les contraintes temporelles ? Donnez son emploi du temps ***(cf. document réponse Figure 4, page 10)***. Vous utiliserez l'ordonnancement EDF.

La secrétaire est assimilée au calculateur dans la mesure où elle est l'exécutant de l'ensemble des opérations du système. L'ensemble de son travail est répartie en diverses tâches.

Sur les chronogrammes des documents réponses, on se souciera donc du temps qui passe et non pas de l'heure proprement dite.

On peut normalement utiliser au choix RM ou EDF, mais comme il y a à priori une contrainte de mise en œuvre, c'est EDF qu'il faudra implémenter. Comme la secrétaire peut suspendre son travail, on prendra un ordonnancement préemptif.

Les différentes tâches considérées dans ce premier exercice :

- *τ_1 - La tâche courrier : r_1=13h, C_1=0,5h, T_1=3h sur la période de travail*
- *τ_2 - L'emploi du temps : r_2=13h, C_2=2,5h, T_2=6h*
- *τ_3 - Le repos : r_3=13h, C_3=0,25h, T_3=2h sur la période de travail*

On peut utiliser l'expression de la condition suffisante pour vérifier la faisabilité de l'ordonnancement en EDF. Pour les tâches de courriers et Emploi du temps (EDT), elles ne s'exécutent qu'en seule fois dans la journée ; on considèrera dans ce cas l'échéance. Pour la tâche repos, celle-ci est périodique et de période 2h.

$$U = \sum_{i=1}^{N} \left(\frac{C_i}{T_i} \right) \leq 1$$

Soit : $U = \sum_{i=1}^{3} \left(\frac{C_i}{T_i} \right) = \frac{C_1}{D_1} + \frac{C_2}{D_2} + \frac{C_3}{D_3} \leq 1$

A.N. : $U = \frac{0,5}{3} + \frac{2,5}{6} + \frac{0,25}{2} = 0,708 \leq 1$ c'est donc ordonnaçable en EDF.

On vérifie sur la figure 4 que le système de tâches est faisable par la secrétaire. On vérifie donc bien le résultat de la condition suffisante.

Pour améliorer ses performances, son supérieur hiérarchique lui propose de ne plus s'interrompre, lorsqu'elle a déjà démarrée une tâche, pour exécuter une autre plus prioritaire.

b) Si l'on considère que le temps pour passer d'un traitement à un autre est négligeable, pensez vous que ce conseil soit judicieux ***(Complétez le document réponse Figure 5, page 10)*** ?

On vérifie bien sur la figure 4 que le système de tâches est faisable par la secrétaire. Ce choix n'est toutefois pas judicieux car cela reviendrait à utiliser un algorithme non préemptif qui vis-à-vis d'un algorithme préemptif est moins efficace. Mais la secrétaire respecte toujours ses échéances. Un bel exemple de réactivité de la part de cette employée.

On prend maintenant en compte l'accueil des étudiants. Les moments d'arrivée des étudiants, leur temps de service et le délai au-delà duquel ils s'impatientent sont données dans le tableau ci-dessous.

Nom de l'étudiant *Tâches apériodiques*	*Temps de service nécessaire (mn)* *Capacité*	*Heure d'arrivée* *Réveil*	*Durée après laquelle l'étudiant s'impatiente (mn)* *Deadline*
Mlle Germaine	15	13h	15
Mlle Laetitia	30	16h	90
Mr Patrick	15	17h45	30

c) Trouvez un ordonnancement (ordonnancement préemptif avec temps de commutation négligeable) qui permette à la secrétaire de respecter ses différentes contraintes de temps tout en servant les étudiants avant qu'ils ne s'impatientent. Votre solution doit absolument garantir le respect des contraintes de temps pour la distribution du courrier, l'élaboration des emplois du temps et les pauses de la secrétaire, quitte le cas échéant à ne pas pouvoir servir les étudiants à temps (et ce, quelque soit la loi d'arrivée des étudiants). Vous justifierez votre solution ***(cf. document réponse Figure 6, page 10)***.

Plusieurs solutions sont possibles ; soit continuer avec l'ordonnancement EDF comme auparavant, soit utiliser un serveur de tâches apériodiques. De ce fait, les tâches périodiques continueront à respecter les contraintes temporelles (tâches courriers, pauses et emploi du temps).

La solution EDF est proposée pour les tâches apériodiques que sont les étudiants (cf. figure 6). → fonctionne, mais ne garantie pas le respect des échéances des tâches périodiques en cas de surcharge.

Une solution utilisant un serveur de tâches par scrutation (cf. figure 7) → cette solution ne fonctionne pas.

Enfin une solution convenable (figure 8) → le serveur sporadique pour les tâches apériodiques.

D. Application multitâche sous linux (10 mn / 3 points)

L'application ci-dessous est fournie sans spécification. Il vous sera demandé d'indiquer ce que fait exactement cette application.

a) Indiquez ce que l'on peut lire sur l'écran de la machine informatique lorsque cette application est exécutée.

Lors de l'exécution de l'application, on pourra lire sur l'écran de la machine informatique :

- *Situation 1 si lorsque le processus père s'exécute, le fils n'existe pas/plus.*
- *Situation 2 si lorsque le processus père s'exécute, le fils existe*

b) Indiquez ce qui pourrait se passer si l'on enlevait les lignes 19, 20, 21, 24.

Si on enlève les lignes indiquées, cela ne change pas grand-chose dans la mesure où le père ayant dormi pendant 5 secondes, le fils a le temps de rentrer dans sa boucle infinie. Le processus fils existant toujours, le signal sera bien reçu par son destinataire. Le résultat est donc toujours identique : le fils est tué par le père qui n'en avait toutefois pas l'intention.

c) Remplacer par des explications simples et claires les commentaires 1 et 2 du programme source qui vous est fourni ainsi que les situations 1 et 2.

```
(1)     #include <stdio.h>
(2)     #include <unistd.h>
(3)     #include <signal.h>
(4)     #include <stdlib.h>
(5)     #include <sys/types.h>
(6)     #include <sys/stat.h>
(7)     #include <errno.h>
(8)     main() {
(10)            pid_t pid;
(11)            int status;
(12)            switch(pid=fork()){
(13)                    case -1:  perror("Création de processus");
(14)                              exit(2); break;
(15)                     case 0:  /*  Commentaire 1 */   _ Code du processus fils_ _ _ _ _ _ _ _ _
(16)                              while (1); break;
(17)                    default:  /* Commentaire 2 */   _ Code du processus père  _ _ _ _ _ _ _ _
(18)                              sleep(5);
(19)                              if (kill(pid,0)== -1)
(20)                                      printf ("Test présence fils : fils n'existe pas !\n");
                                          // test de la  présence du processus fils
(21)                              else {
(22)                                      printf ("Père envoi SIGUSR1 vers le fils ! \n");
                                          // Le père envoie le signal SIGUSR1 au fils.
                                          // Comme ce signal n'a pas été dérouté par le fils,
                                           // Le comportement par défaut est SIGKILL !
                                           // le processus fils  sera donc tué par  le processus père !

(23)                                      kill(pid,SIGUSR1);
(24)                              }
(25)            }
(26)    }
```

τ_1 (r_1=0,C_1=7,T_1= 29)
τ_2 (r_2=0,C_2=1,T_2= 5)
τ_3 (r_3=0,C_3=2,T_3= 10)

Document réponses

Ordonnanceur **non-préemptif**

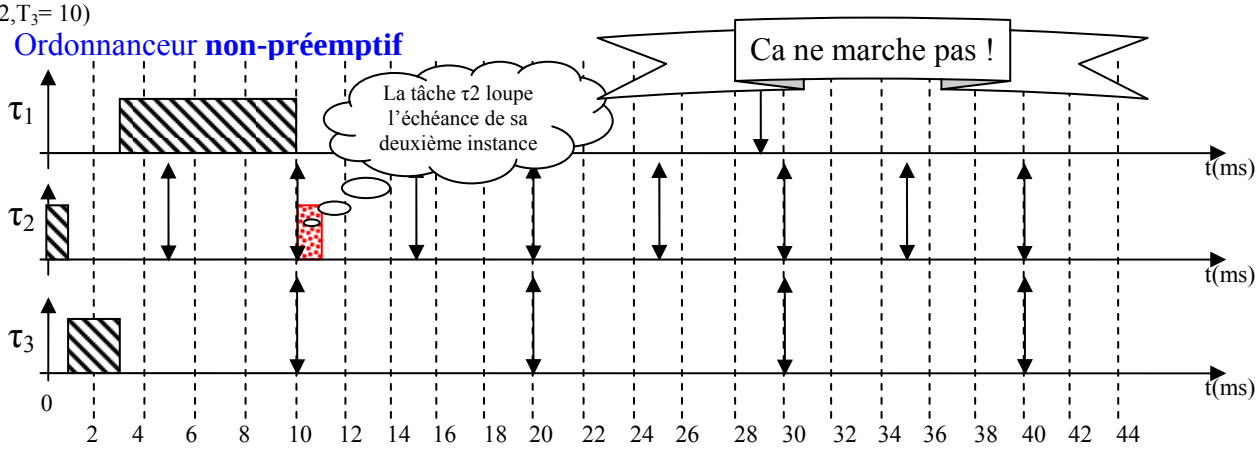
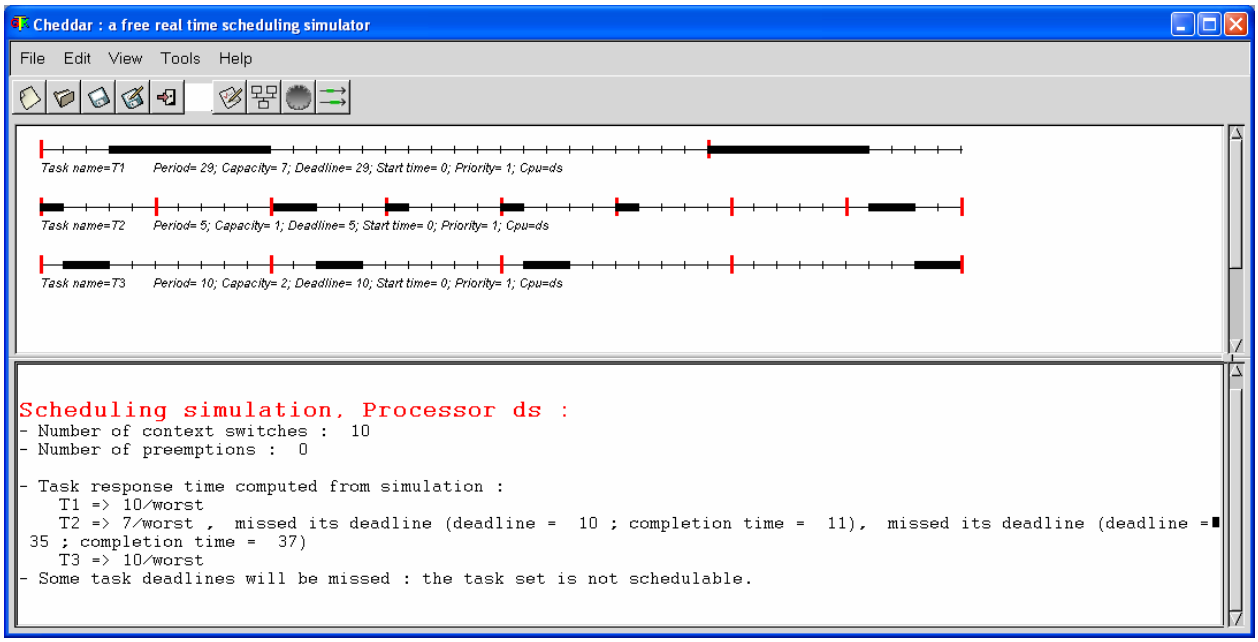


Figure 1 - B-2 e) Ordonnancement « Rate Monotonic » non préemptif



B-2 e) La correction version Cheddar, qui vous donne toutes les informations nécessaires !

τ_1 (r_1=0,C_1=7,T_1= 29)
τ_2 (r_2=0,C_2=1,T_2= 5)
τ_3 (r_3=0,C_3=2,T_3= 10)

Ordonnanceur **préemptif**

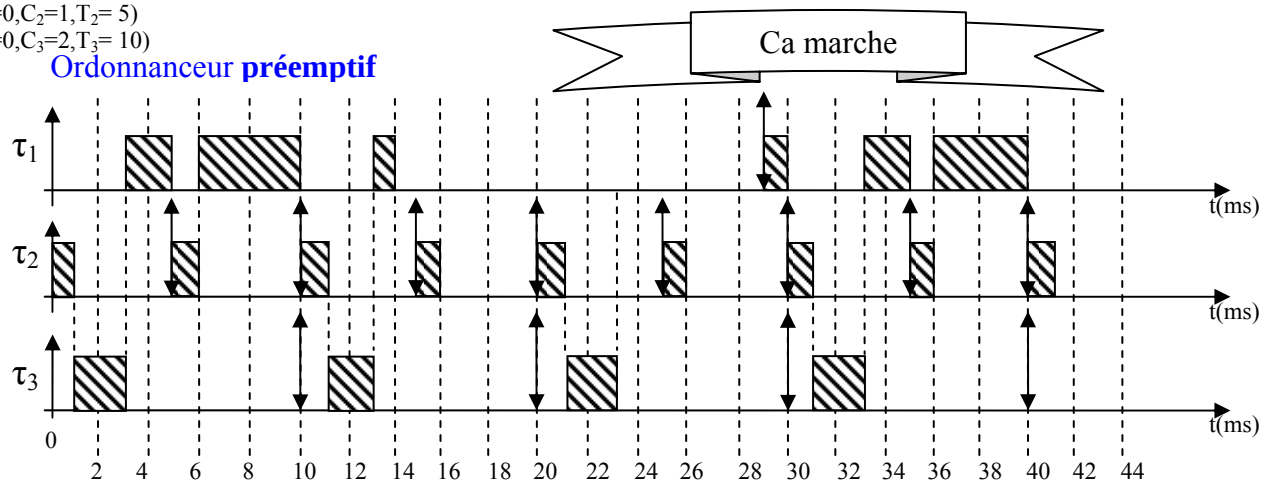
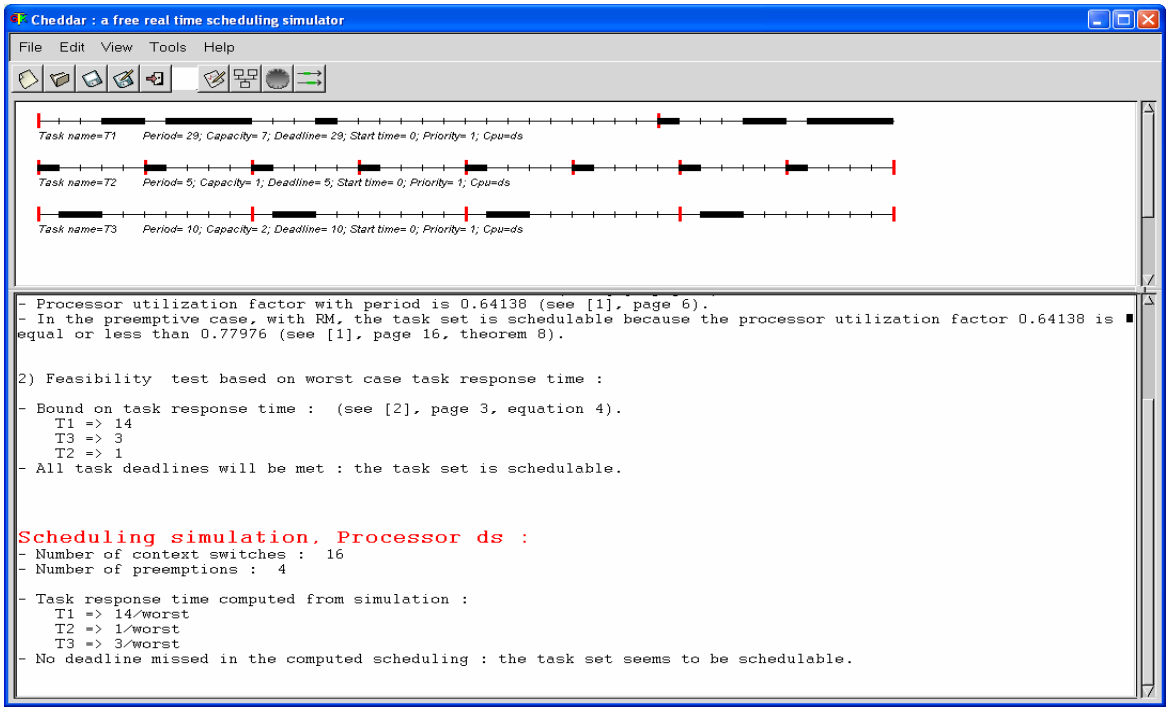


Figure 2 - B-2 e) Ordonnancement « Rate Monotonic » préemptif



B-2 e) La correction version Cheddar, qui vous donne toutes les informations nécessaires !

τ_1 (r_1=0,C_1=6,T_1= 30)
τ_2 (r_2=0,C_2=3,T_2= 5)
τ_3 (r_3=0,C_3=2,T_3= 10)

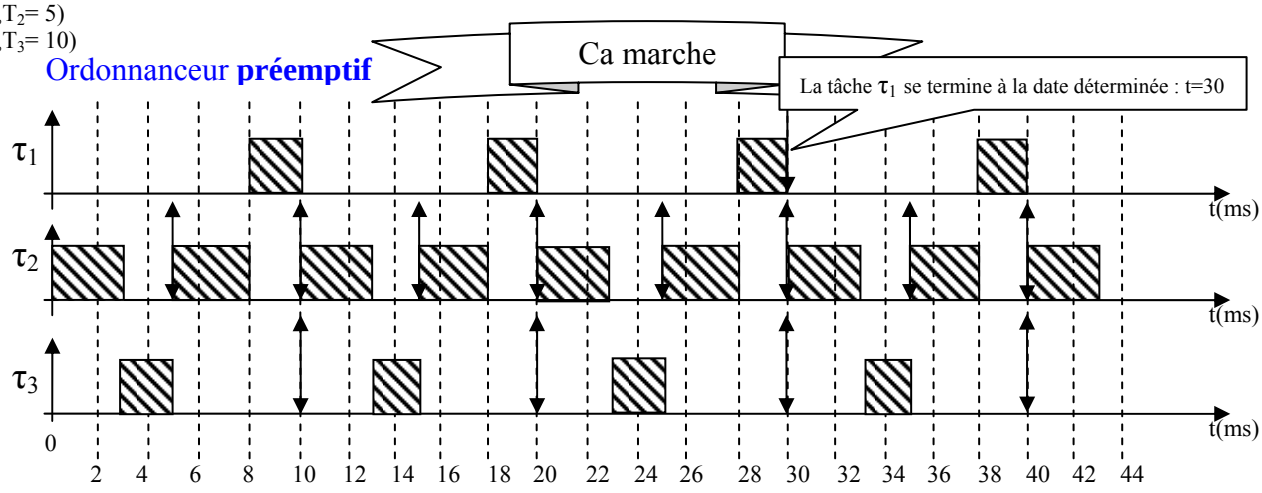
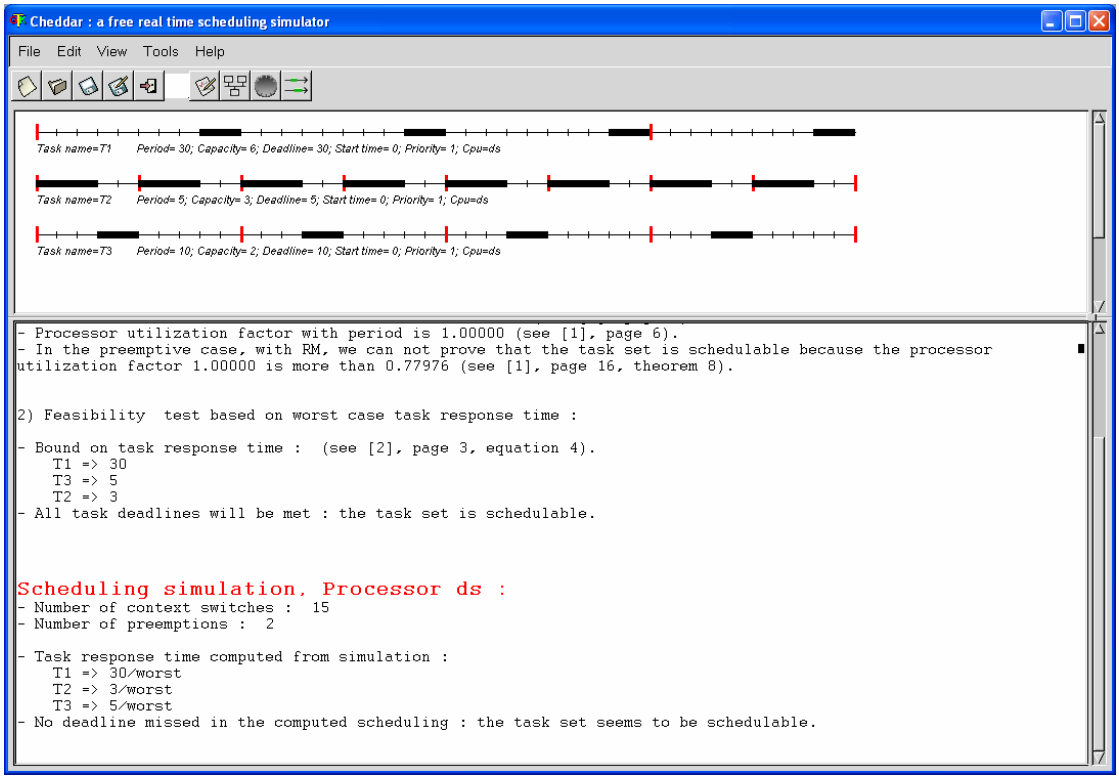


Figure 3 - B-2 f) Ordonnancement « Rate Monotonic » préemptif

Remarque : dans le cas d'une égalité, on privilégie la dernière tâche active ; dans un autre cas, on privilégiera la tâche qui est la plus vieille dans la file des tâches réactivées



La correction version Cheddar, qui vous donne toutes les informations nécessaires !

τ_1 (r_1=13h, C_1=0.5h, T_1= 3h)
τ_2 (r_2=13h, C_2=2.5h, T_2= 6h)
τ_3 (r_3=13h, C_3=0.25h, T_3= 2h)

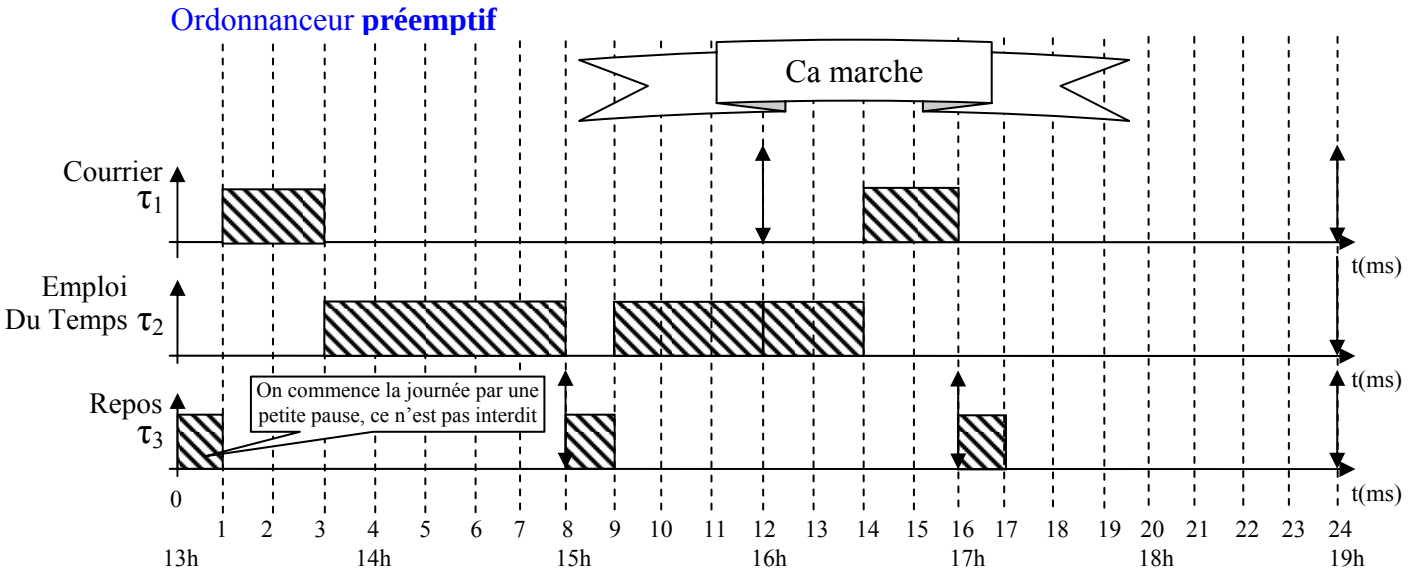
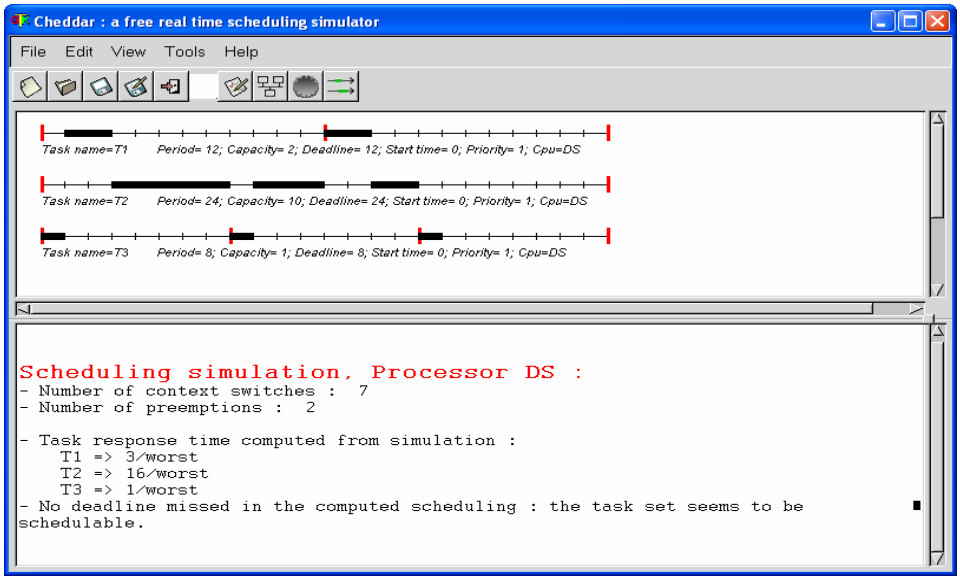


Figure 4 – C a) Emploi du temps de la secrétaire avec un ordonnancement EDF préemptif



C a) La correction version Cheddar, qui vous donne toutes les informations nécessaires !

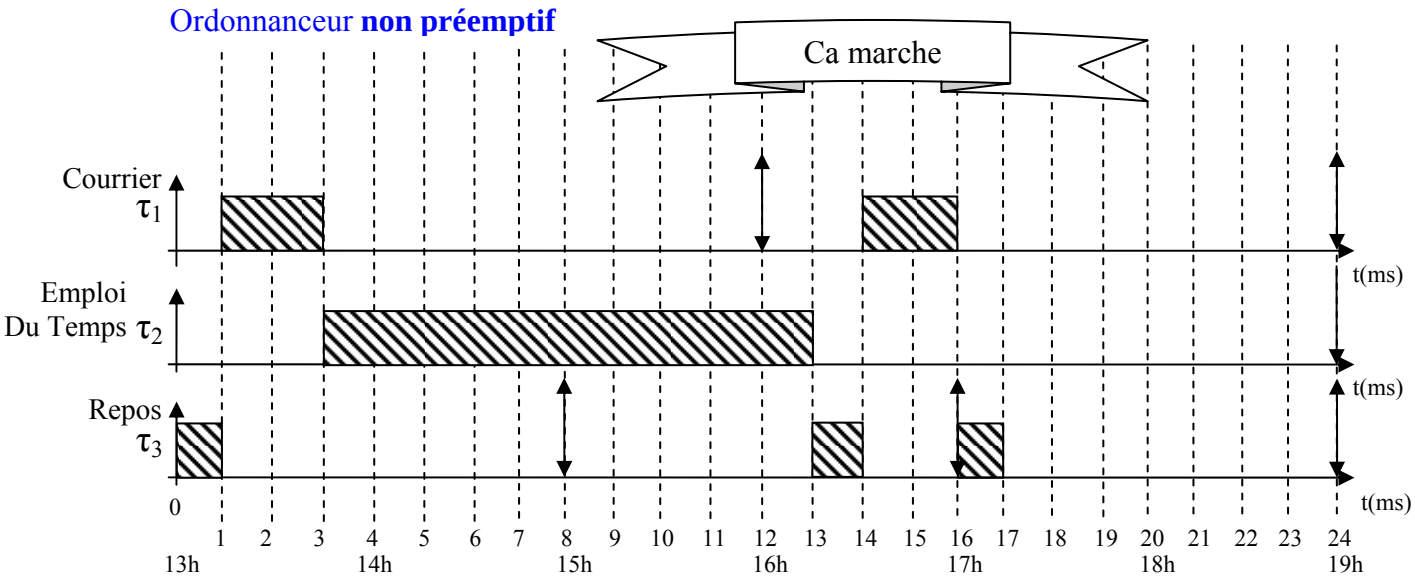
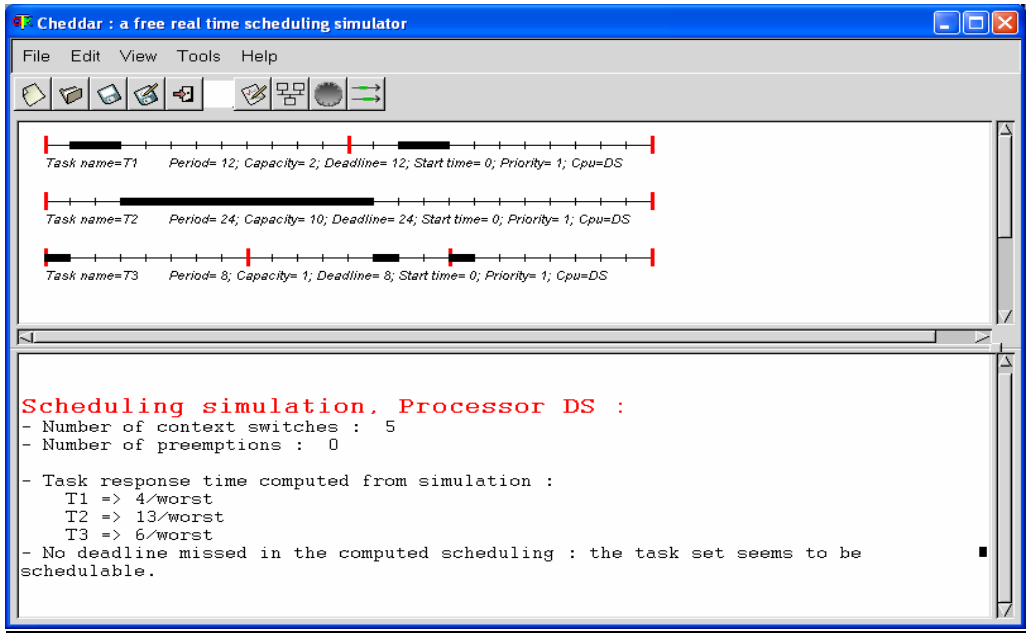


Figure 5 – C b) Emploi du temps de la secrétaire avec un ordonnancement EDF non préemptif



C b) La correction version Cheddar, qui vous donne toutes les informations nécessaires !

Remarque : dans le cas d'une égalité, on privilégie la dernière tâche active ; dans un autre cas, on privilégiera la tâche qui est la plus vieille dans la file des tâches réactivées

Si on rend la tâche τ_1 la moins prioritaire, elle ne respecte plus son échéance → si on cherche à éviter qu'elle ne commence dès le début de la journée.

Pour bien voir l'effet de la pause, il faudrait une granularité plus fine de l'ordonnanceur, mais l'efficacité s'en trouverait surement amoindrie dans la mesure où faire plusieurs choses à la fois nécessite un grand nombre de changement de contexte. Ne dit-on pas courir deux lièvres à la fois.

Remarques personnelles :

- *De plus, si les pauses sont non préemptives, on ne pourra plus découper celles-ci. Si par malheur un petit besoin se faisait sentir, il faudrait attendre.*
- *On voit là un problème du respect formel des règles sur le terrain. On occulte complètement le côté de l'environnement de travail. Il faut à mon avis rester près de son personnel, il en sera sûrement plus efficace et plus motivé.*

Pour voir ce que cela fait avec un ordonnancement EDF préemptif !

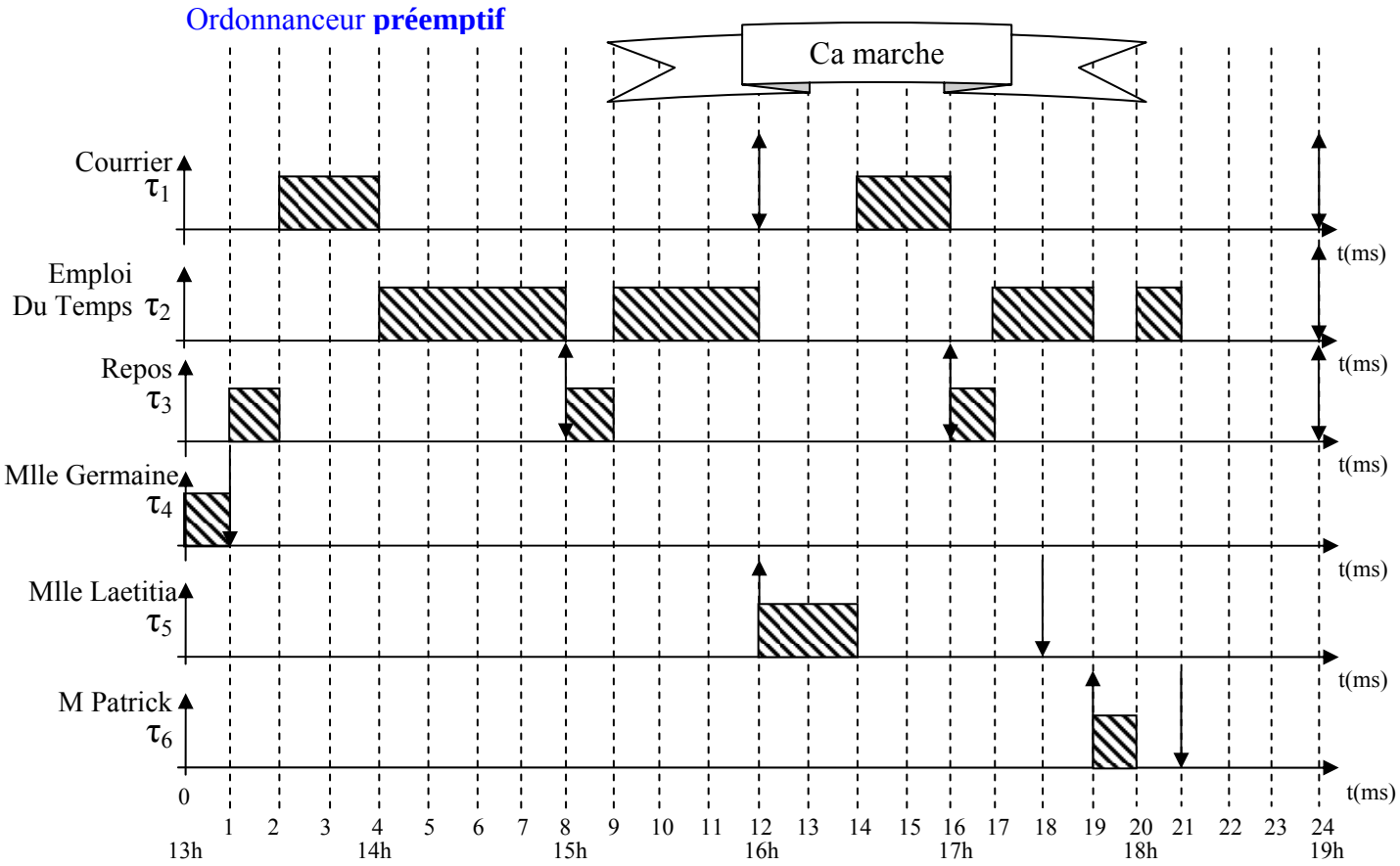


Figure 6 – C c) Emploi du temps de la secrétaire avec un ordonnancement EDF préemptif

Remarque1 : chacune des unités de cet ensemble de diagrammes correspond à 15 minutes (1/4 d'heure).

Remarque2 : dans le cas d'une égalité, on privilégie la dernière tâche active ; dans un autre cas, on privilégiera la tâche qui est la plus vieille dans la file des tâches réactivées

On remarque que les tâches représentatives des étudiants ayant globalement, une fois réveillées, des échéances plus proches que les autres tâches, sont exécutées de façon très prioritaire. Les tâches « courrier », « EDT », « repos » peuvent dans ce cas là ne pas respecter leur échéance à cause l'intervention des étudiants.

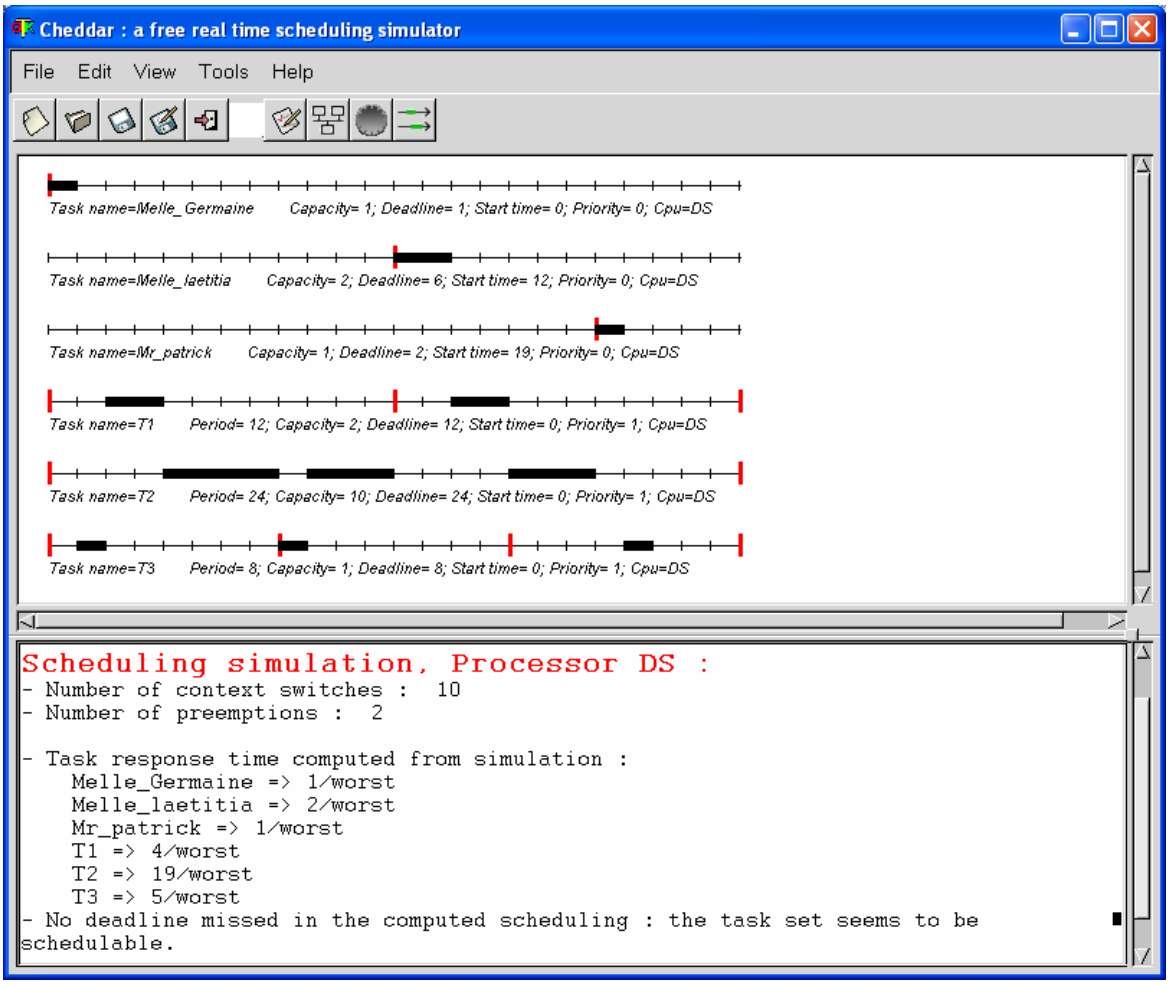
Cette solution ne permet donc pas de répondre complètement au cahier des charges.

On fait évoluer cette solution en utilisant un serveur de tâches apériodiques. Celui-ci est invoqué périodiquement et servira les tâches apériodiques qui auront été activées, mais seulement lorsque qu'un créneau de temps lui sera alloué. Cela permet de garantir le traitement des tâches périodiques (elles respecteront leur échéance quelque soit

le nombre de tâches apériodiques dans le système) : pas de surcharge ramenée par le traitement des tâches apériodiques.

Je vous propose un serveur de scrutation :

- *Quand il est activé, le serveur exécute les tâches en attente jusqu'à épuisement de celles-ci ou de sa capacité.*
- *Dès qu'il n'y a plus de tâches en attente, il se suspend jusqu'à sa prochaine exécution périodique.*
- *Sa capacité est réinitialisée à chaque nouvelle exécution.*



C c) La correction version Cheddar, qui vous donne toutes les informations nécessaires !

Il n'y a plus de solution proposée avec Cheddar, celui-ci ne permettant (à mon avis) pas la mise en œuvre de serveur de tâches apériodiques.

The reader can find information and tutorial on the author website: http://beru.univ-brest.fr/~singhoff/cheddar/ in English and French version.

We will probably propose a solution to this lack in the future with the help of the ECE Students.

Ordonnancement avec serveur de tâches apériodiques et ordonnanceur de tâches périodiques préemptif et EDF !

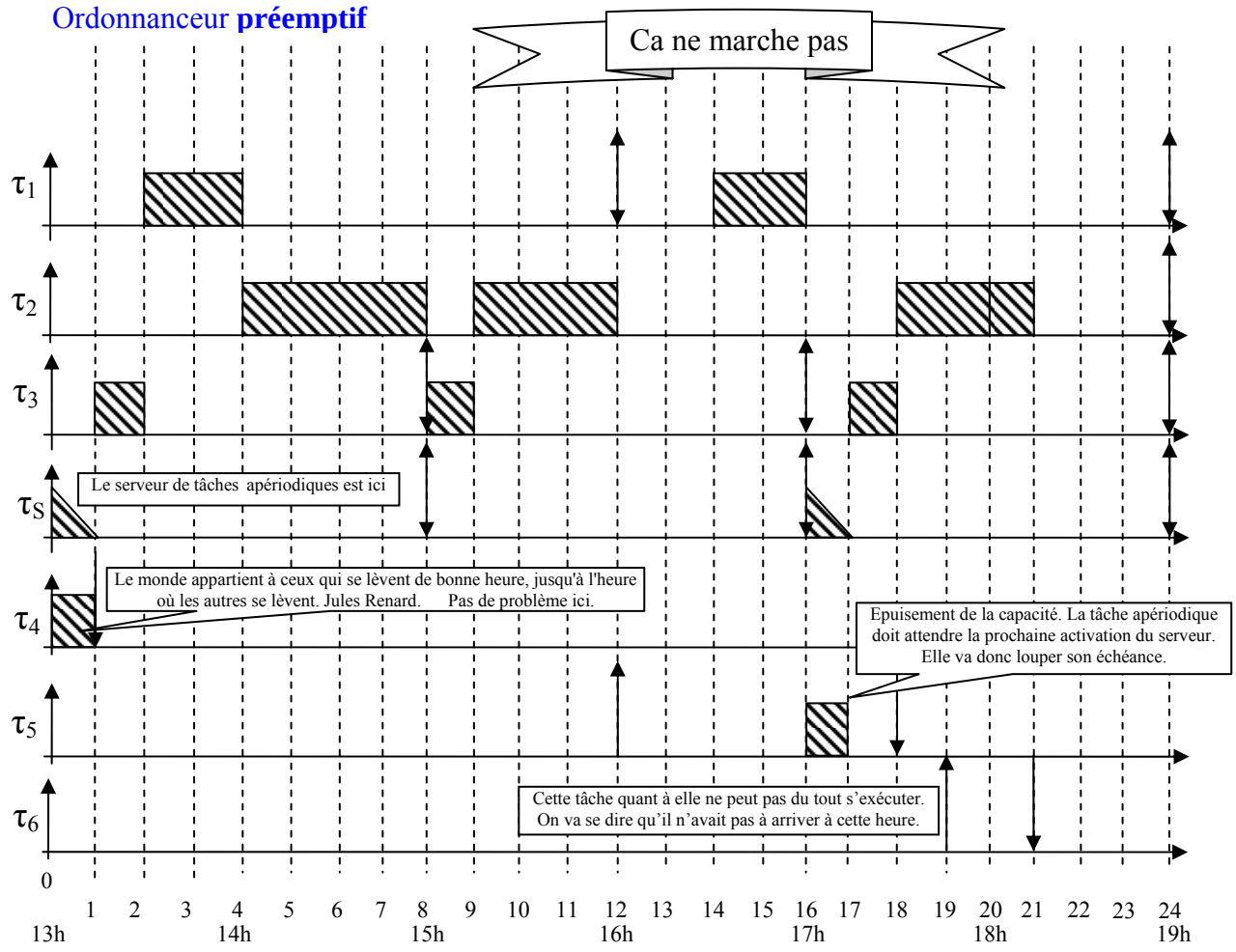


Figure 7 – C c) Emploi du temps de la secrétaire avec un serveur par scrutation et EDF préemptif

On cherche à définir les paramètres du serveur de tâches apériodiques qui permettra aux tâches périodiques initiales de respecter leur échéance.

On considère que la journée ne commence pas par une pause. On privilégie l'accueil des étudiants par rapport à la tâche repos : dans le bus de ne pas commencer la journée par une pause (car ces deux tâches ont la même priorité).

On peut considérer en regardant la séquence EDF tracée précédemment, une capacité de 15 minutes toutes les deux heures pour le service aux étudiants. Réveil de la première instance à 13h et échéance sur requête du serveur.

Vérification de l'ordonnançabilité à l'aide de la condition suffisante de Liu & Leyland :

$$U = \sum_{i=1}^{4} \left(\frac{C_i}{T_i} \right) = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} + \frac{C_4}{T_4} \leq 4(2^{\frac{1}{4}} - 1)$$

A.N. : $U = \frac{0,5}{3} + \frac{2,5}{6} + \frac{0,25}{2} + \frac{0,25}{2} \leq 4(2^{\frac{1}{4}} - 1)$

$U = 0,833 \leq 0,756$ *La condition suffisante n'est pas vérifiée, il faudra tester le respect de chacune des échéances. Dans notre cas, on va simplement tracer la séquence pour cet ensemble de tâche.*

On remarque que le temps alloué au serveur de scrutation est insuffisant pour servir les tâches apériodiques.

On peut donc envisager les solutions :

- *D'augmenter la capacité du serveur de scrutation : cela permettra à la tâche « Mle Laetitia » de terminer sa requête. Par contre Mr Patrick ne peut toujours pas être servi.*
- *du serveur sporadique :*

Rappel : Le serveur sporadique est un type de serveur, permettant d'améliorer le temps de réponse des tâches apériodiques, ***sans cependant diminuer le taux d'utilisation du processeur des tâches périodiques****.*
Le serveur sporadique s'exécute chaque fois qu'il y a une demande de tâche sporadique ***et*** *qu'il a une capacité non nulle* ***et*** *que sa priorité est éligible (pas de tâche plus prioritaire).*
Il s'arrête dès qu'il a épuisé sa capacité ou servi toutes les commandes en attente.
Une certaine capacité lui est restituée, une période après le début de son exécution.

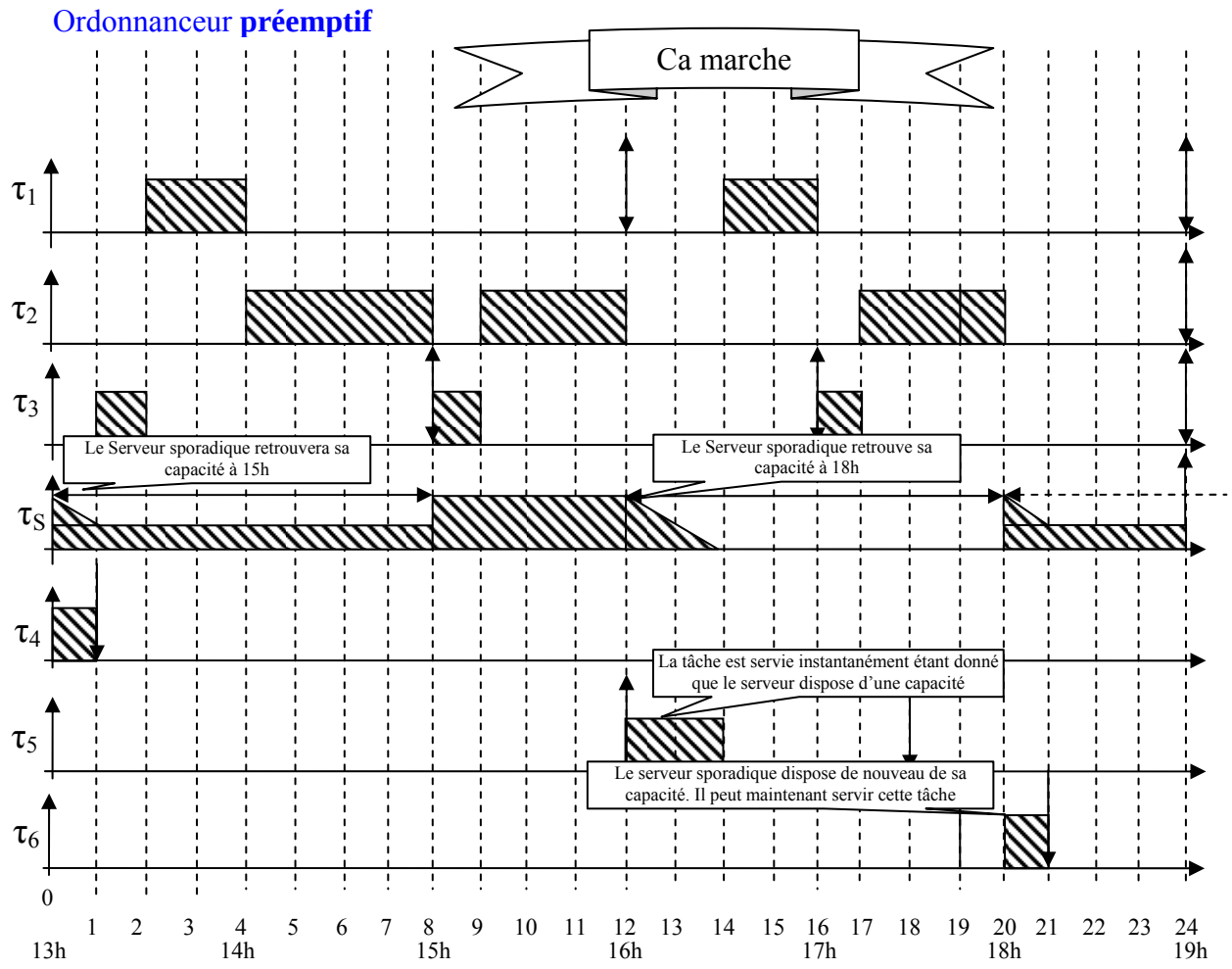


Figure 8 – C c) Emploi du temps de la secrétaire avec un serveur sporadique et EDF préemptif

Le serveur sporadique démarre à 13h et à une période de 2h. On lui alloue une capacité de 30 minutes à chaque réinitialisation.

On a dessiné pour τ_S la capacité restante du serveur ayant en charge la gestion des tâches apériodiques.

Avec ce type de serveur, l'ensemble des tâches respectent leur échéance : les tâches périodiques et les tâches apériodiques. On peut également avancer que les tâches périodiques respecteront leur échéance même en cas de surcharge dues à l'arrivée d'un grand nombre d'étudiants.

Il faut voir maintenant la façon de former la secrétaire pour la mise en œuvre de cette méthode de gestion des activités. Ce n'est sûrement pas avec un programme informatique qu'on va lui dire de s'arrêter. Ce sera à vous de juger lorsque vous serez confronté à ce genre de problèmes dans votre vie active.