

1 - MULTI-PROGRAMMATION

1. INTRODUCTION

Un *programme* est l'expression d'un algorithme à l'aide d'un langage de programmation.

Une *tâche* est la conjonction d'un programme et des données auxquelles il s'applique.

Un *processus* est l'exécution d'une tâche. Il est caractérisé par la tâche et son contexte d'exécution, c'est-à-dire la valeur du compteur ordinal, les valeurs des registres etc...

En *monoprogrammation*, il y a un seul processus à la fois en mémoire. Lorsqu'une tâche est soumise et que le processeur est disponible, on la charge en mémoire puis on exécute le processus associé jusqu'à ce qu'il soit terminé. On passe alors à la tâche suivante.

En *multiprogrammation*, il peut y avoir plusieurs processus à la fois en mémoire. Une tâche soumise est chargée en mémoire s'il y a de la place et donne naissance à un processus. Un processus est prêt s'il n'est pas en attente d'une entrée-sortie. Les processus prêts s'exécutent à tour de rôle, on parle de commutation de processus. La multiprogrammation peut être utilisée en batch ou en temps partagé.

En *batch*, il n'y a commutation que si le processus actif doit effectuer une entrée-sortie.

En *temps partagé*, il y a commutation si le processus actif doit effectuer une entrée-sortie ou s'il a épuisé son quantum.

Un *quantum* est une durée élémentaire (de l'ordre de 10 à 100 ms).

L'*overhead* est le temps passé (perdu) à effectuer une commutation entre deux processus en temps partagé.

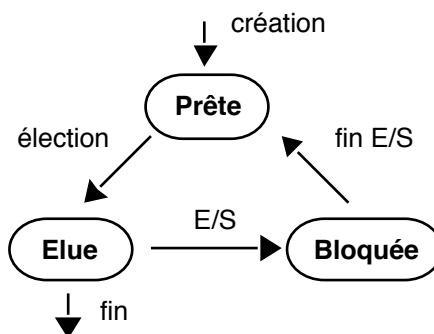
Une *unité d'échange* sert à effectuer des transferts entre la mémoire principale et un périphérique sans utiliser le processeur (DMA). Elle reçoit un ordre du processeur, effectue le transfert, puis avise le processeur de la fin de l'échange. Durant le transfert, le processeur peut faire d'autres calculs.

On se propose d'étudier analytiquement l'influence des différentes politiques de gestion d'un processeur. On considère un ordinateur doté d'un processeur principal et d'une unité d'échange effectuant les entrées-sorties. On suppose qu'il y a suffisamment de mémoire pour contenir tous les processus.

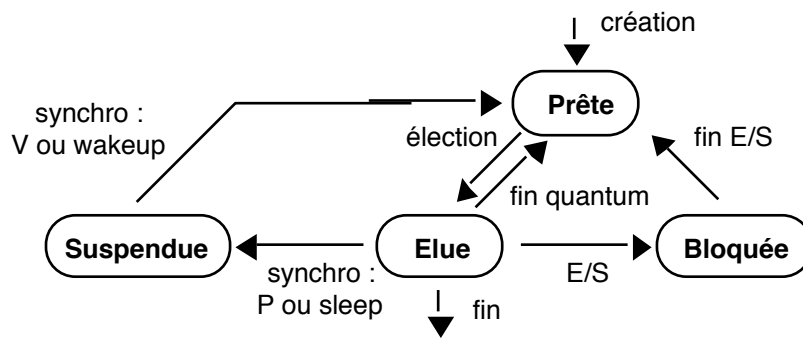
1.1.

Représentez sous forme de diagramme d'états (noeuds = états d'une tâche, arcs = actions ou événements) l'exécution d'une tâche en mode batch puis en temps partagé.

Exécution en mode batch :



Exécution en temps partagé :



2. ETUDE DU TAUX D'OCCUPATION

On s'intéresse à l'exécution de trois tâches T_1 , T_2 et T_3 :

T_1 dure 200 ms (hors entrée/sortie) et réalise une unique E/S au bout de 110 ms ;

T_2 dure 50 ms et réalise une unique entrée/sortie au bout de 5 ms ;

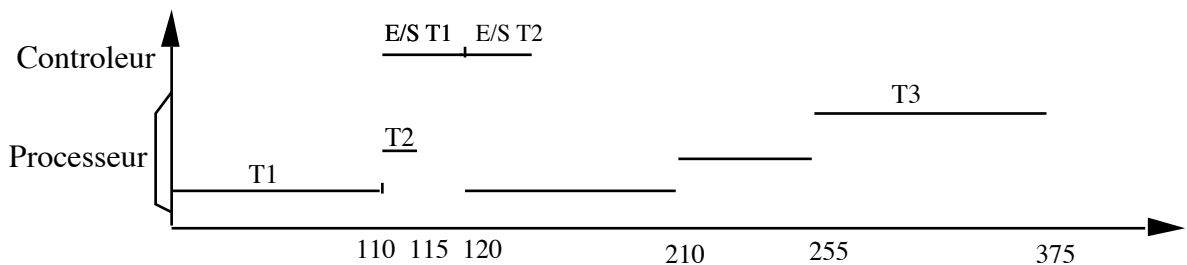
T_3 dure 120 ms sans faire d'entrée/sortie

T_1 et T_2 sont créées à l'instant 0, T_3 est créée à l'instant 140. Chaque entrée/sortie dure 10 ms. La valeur du quantum (pour le temps partagé uniquement) est de 100 ms.

2.1.

Représentez sur un diagramme des temps l'exécution des 3 tâches en mode batch et en temps partagé. Calculez le temps total d'exécution T et le taux d'occupation du processeur τ_p .

En mode batch :

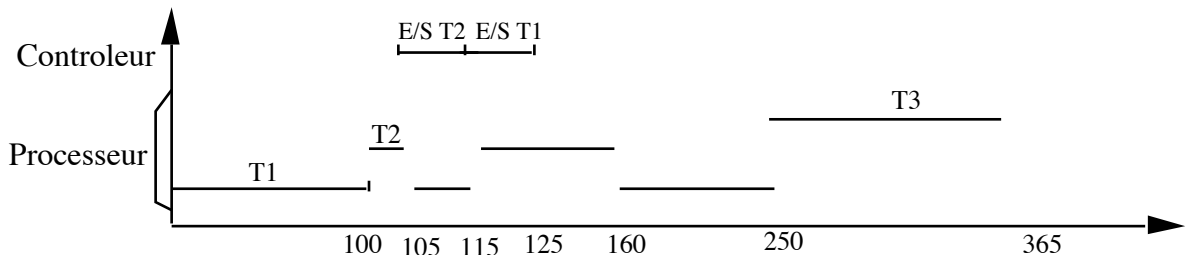


Rem 1 : On suppose ici qu'il n'y a pas de réquisition

Rem 2 : à $t = 210$, les tâches T_2 et T_3 sont prêtes. T_2 est prête depuis $t = 130$ alors que T_3 n'est prête que depuis $t = 140$. On choisit donc d'exécuter T_2 .

Taux d'occupation = $370/375$ (temps cumulé d'exécution des tâches/temps total) = 98,7 %

En temps partagé :



On considère maintenant deux tâches identiques T_1 et T_2 effectuant n fois le traitement suivant :

- Lecture d'une donnée sur le disque (durée 1)

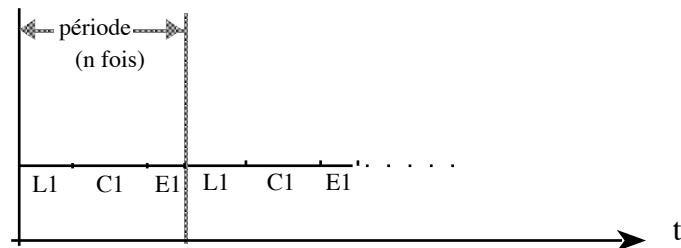
- Opération de calcul sur la donnée (durée c)
- Ecriture du résultat sur le disque (durée e)

2.2.

La tâche T_1 est soumise seule à $t = 0$. Représentez sur un diagramme de temps l'exécution de la tâche T_1 . Calculez le temps total d'exécution T , le taux d'occupation du processeur τ_P et de l'unité d'échange τ_U .

Que peut-on en déduire pour T , τ_P et τ_U lorsque les tâches T_1 et T_2 sont exécutées en monoprogrammation ? Application numérique : $l = e = 7$, $c = 6$, $n = 4$.

La tâche T_1 est soumise seule :



Temps total d'exécution T : $T = n \cdot (l + c + e)$

Taux d'occupation du processeur τ_P : $\tau_P = =$

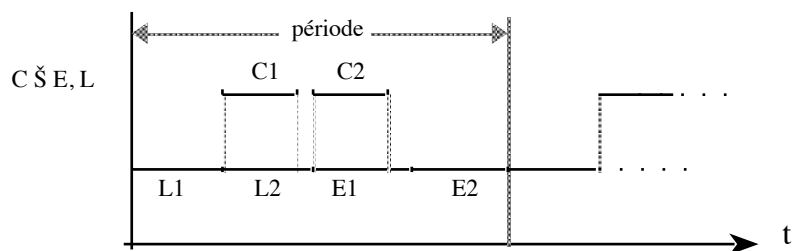
Taux d'occupation de l'unité d'échange τ_U : $\tau_U = =$

Lorsque T_1 et T_2 sont exécutées en monoprogrammation, l'exécution de T_1 est suivie de l'exécution de T_2 qui est exactement identique. Le temps total d'exécution est donc doublé. Par contre, τ_P et τ_U ne sont pas modifiés puisque le temps d'exécution sur le processeur et sur l'unité d'échange sont doublés aussi.

Application numérique : $T = 160 \text{ ms}$ $\tau_P = 30\%$ $\tau_U = 70\%$

2.3.

Représentez sur un diagramme des temps l'exécution des tâches T_1 et T_2 en supposant maintenant une multiprogrammation en mode batch. On supposera $c \leq l$ et $c \leq e$. Calculez T , τ_P et τ_U pour les mêmes valeurs numériques qu'à la question précédente.



$T = 2 \cdot n \cdot (l + e)$

$\tau_P = =$

$\tau_U = 1$

Application numérique : $T = 112 \text{ ms}$ $\tau_P = 43\%$ $\tau_U = 100\%$

2.4.

Y a-t-il un intérêt à la multiprogrammation en mode batch si l'ordinateur ne dispose pas d'une unité d'échange?

Grâce à l'unité d'échange, la multiprogrammation permet d'améliorer les taux d'occupation. Sans unité d'échange, il n'y aurait pas de parallélisme, donc aucun intérêt à la multiprogrammation.

3. TEMPS DE REPONSE EN MODE BATCH

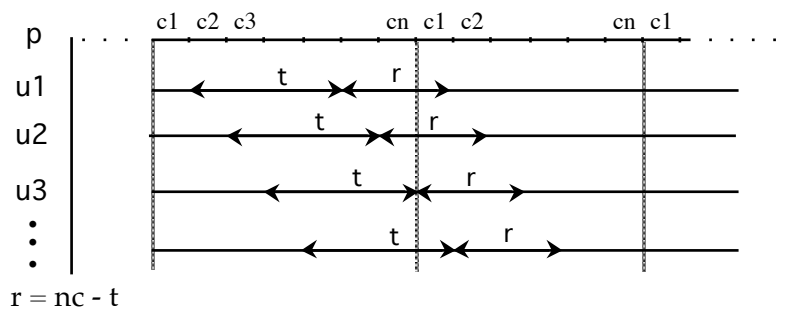
On suppose qu'il y a n utilisateurs exécutant des commandes d'édition. Chaque commande consomme c secondes de temps processeur. Chaque utilisateur attend r secondes (temps de réponse) entre l'instant où il soumet une commande et l'instant où il obtient la réponse, il s'écoule ensuite t secondes (réflexion et frappe) avant qu'il soumette la commande suivante.

3.1.

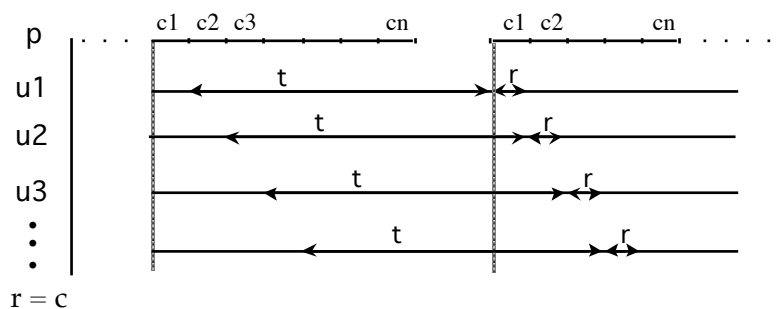
Tous les utilisateurs commencent à travailler en même temps. Exprimez le temps de réponse en fonction des autres paramètres en régime permanent. Calculez r pour $n = 10$, $t = 2$ et $c = 0,1$; puis pour $n = 30$, $t = 2$ et $c = 0,1$.

On doit considérer 2 cas, suivant que le temps de réflexion d'un utilisateur est inférieur ou supérieur au temps nécessaire au processeur pour traiter les commandes des $(n - 1)$ autres utilisateurs.

- $t \leq (n - 1) c$



- $t \geq (n - 1) c$



Application numérique :

1/ $(n - 1) c = 0.9 < t$ donc $r = 0.1$ ms

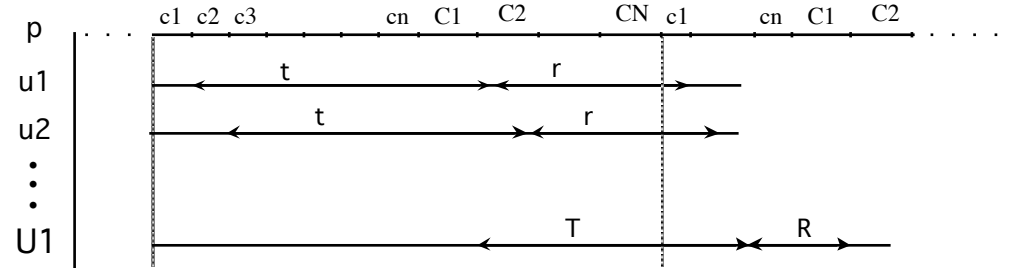
2/ $(n - 1) c = 2.9 > t$ donc $r = 1$ ms

On suppose qu'il y a une autre classe d'utilisateurs effectuant des compilations. On note N , C , R , T les paramètres correspondant à cette classe. On suppose $t = T$, $t \leq (n-1)c + NC$ et $T \leq nc + (N-1)C$.

3.2.

Tous les utilisateurs commencent à travailler en même temps. Exprimez les temps de réponse r et R en fonction des autres paramètres en régime permanent. Montrez que $r = R$. Calculez r pour $n = 10$, $t = 2$, $c = 0,1$, $N = 5$ et $C = 3$.

On a



$$t + r = nc + NC \quad T + R = nc + NC$$

$$\text{donc } r = R = nc + NC - t$$

Application numérique : $r = 1 + 15 - 2 = 14$ ms

3.3.

Est-il raisonnable d'envisager un travail interactif en mode batch?

Un travail interactif, tel l'édition, n'est pas envisageable en mode batch dès lors que des gros travaux sont soumis : on voit en effet que le temps de réponse augmente considérablement !

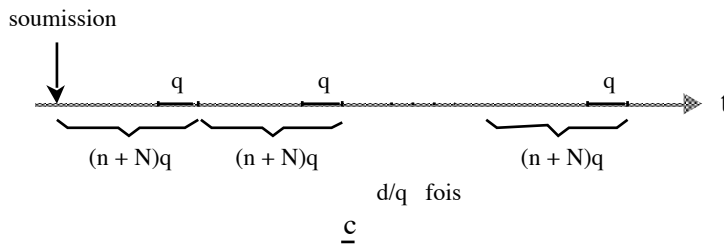
4. TEMPS DE REPONSE EN TEMPS PARTAGE.

Dans cette partie, on note q le quantum et on suppose qu'il n'y a pas d'overhead. On considère à nouveau n utilisateurs effectuant des commandes d'édition et N utilisateurs effectuant des compilations. Pour simplifier l'analyse, on suppose que $t = T = 0$ et que c et C sont multiples de q .

4.1.

Tous les utilisateurs commencent à travailler en même temps. Exprimez les temps de réponse r et R en fonction des autres paramètres en régime permanent. Calculez r et R pour $n = 10$, $c = 0,1$, $N = 5$ et $C = 3$.

Comme $t = T = 0$, le nombre de processus prêts est constant, égal à $(n + N)$. Chaque processus obtient donc un quantum toutes les $(n + N)q$ millisecondes. Un processus de durée d va donc s'exécuter comme suit :



$$\text{On a donc } r = (n + N)q \cdot \frac{c}{q} = (n + N)c \text{ et } R = (n + N)C$$

Application numérique : $r = 1,5$ ms $R = 45$ ms

4.2.

Conclusion ?