Tubes

Points à connaitre:

1) read et write permettent de lire et d'écrire dans un tube de communication. Par défaut, les lectures et les écritures dans un tube de communication sont bloquantes.

2) Le nombre de lecteurs du tube (resp. écrivains) est le nombre de descripteurs en lecture
(resp. en écriture) associés au tube. La fin de fichier est atteinte si le tube est vide et le
nombre d'écrivains est 0

3) Un tube anonyme (|) est considéré comme un fichier temporaire permettant d'établir une communication unidirectionnelle entre le
créateur du tube et ses descendants. Lorsque tous les descripteurs de fichiers associés à ce tube sont fermés, le
tube est détruit.

4) Les numéros des descripteurs de fichiers créés sont récupérés dans le tableau fd[2] :
- fd[0] ! le numéro du descripteur à utiliser pour lire du tube : read(fd[0], ….); - fd[1] ! le numéro du descripteur à utiliser pour écrire dans le tube : write(fd[1], …);

5) Il est possible d'établir une communication bidirectionnelle père-fils en créant deux tubes anonymes (un pour chaque sens de communication). Ces deux tubes doivent être créés avant la création du fils.

6) L'appel système int mkfifo(const char* path, mode_t mode) permet de créer un tube nommé permettant de communiquer entre les processus d'une même machine.

7) Dans un tube nommé on doit toujours ouvrir en readonly et writeonly pour assurer qu'il y est minimalement un lecteur et un écrivain.

8) Par défaut, l'entrée standard d'un processus est associée au clavier. Les sorties standard et erreur sont associées au moniteur. dup2(fd,2) pour réassigner l'entrée standard ou la sortie. fd[entré, sortie] pour les index.

ATTENTION : Lorsque plusieurs tubes nommés sont utilisés, les ouvertures de ces tubes dans des ordres différents peuvent causer un interblocage.

Exclusion mutuelle

Quatre conditions sont nécessaires pour réaliser correctement une exclusion mutuelle :

1. Deux processus ne peuvent pas être, en même temps, dans leurs sections critiques.

2. Aucune hypothèse ne doit être posée sur les vitesses relatives des processus, le nombre de processeurs, etc.

3. Aucun processus en dehors de sa section critique ne doit bloquer les autres processus d'entrer dans leurs sections critiques.

4. Aucun processus ne doit attendre trop longtemps avant d'entrer en section critique (attente bornée).

Rappel:

- Une fonction atomique est accède toujours en exclusion mutuelle aux données

Monitors

Concept:

Le monitor est une structure de données composé de méthode qui seront accédé en exclusion mutuel. Au lieu d'avoir des sémaphores pour assurer l'exclusion mutuel des parties critiques, les processus sont mis dans une file d'attente FIFO permettant d'exécuter les processus en exclusion mutuelles.

Conditions:

Pour éviter des attentes infinies, des variables de conditions ayant une fonction wait et signal sont utilisées pour contrôler l'exécution dans la file.

Signaux

Tables dans les processus:

1) Table des gestionnaires de signaux (TGS) qui indique pour chaque signal le traitement associé.

2) Table de bits des signaux en attente (TSA) qui indique les signaux reçus mais non encore

3) Table de bits des signaux à différer (MASK) ➔ masque des signaux du processus.

Signaux populaires:

- SIGPIPE : communication tube rompue

- SIGCHiLD : processus enfant terminé

- SIGSEGV : segmentation fault

Contrôle du masque avec :

- SIG_BLOCK : pour ajouter les signaux de set au masque.

- SIG_UNBLOCK: pour retirer les signaux de set du masque.

- SIG_SETMASK: pour remplacer le masque courant par set

Les signaux SIGKILL et SIGSTOP ne peuvent pas figurer dans le masque.

Verrou actif

Utilise une fonction atomique pour contrôler l'accès aux ressources.

Problèmes:

- Attente active ➔ consommation du temps CPU.

- Durée de l'attente active dépendante de celles des sections critiques.

- Risque de boucle infinie si l'ordonnancement est préemptif et à priorités fixes.

Sémaphores

Les sémaphores sont manipulés au moyen d'opérations atomiques :

- **P()** (down ou wait) pour demander un jeton (décrémente si ressource en haut de 0)

- **V()** (up ou signal) pour libérer un jeton. (incrémente aucun processus en attente de la ressource)

Types de sémaphores:

- FIFO (First In First Out) -> Sémaphore fort (par défaut)

- LIFO (Last In First Out) -> Sémaphore faible.

ATTENTION : les sémaphores faibles offrent un risque de famine

Rappel : un mutex est un sémaphore binaire (1 ressource max)

Algorithme du Banquier

Basé sur la formule : A = E - (Alloc(P1) + …. + Alloc(Pn)) ou A, E, Alloc sont les ressources dispos, allouées + dispos, allouées par process.

Souvent on part de Req qui est le tableau des ressources potentiellement nécessaires à chaque processus mais non encore détenues et on va chercher le premier process qui peut fit dans A. Une fois trouver alloue le process (incrémente A) et on refait le tour de Req. Si tout les process rentrent, alors on obtient un état sur.

Interblocages

Conditions de Coffman:

1) Exclusion mutuelle : Une ressource est soit allouée à un seul processus, soit disponible.

2) Détention et attente : Les processus qui détiennent des ressources peuvent en demander d'autres.

3) Pas de réquisition : Une ressource allouée est uniquement libérée par le processus qui la détient (ressources non-préemptives).

4) Attente circulaire : Chaque processus d'un ensemble est en attente d'une ressource détenue par un autre processus de ce même ensemble.

Solutions:

1) Détection des interblocages et reprise à l'aide d'un graphe d'allocation des ressources.

2) Évitement des interblocages en utilisant l'algorithme du banquier.

3) Prévention des interblocages en s'assurant qu'au moins l'une des quatre conditions nécessaires à l'interblocage n'est jamais satisfaite.

Mémoire virtuelle

Types:

- **Pagination pure**: La mémoire est divisé en page et cadre de même taille. Les pages de l'espace d'adressage peuvent donc être chargées dans des cadres (mémoire physique) à l'aide d'adresse virtuelle.

- **Segmentation pure**: On divise la mémoire en segment de taille variable et on utilise la table des segments pour trouver l'adresse d'un segment.

- **Combiné**: L'espace d'adressage d'un processus est un ensemble de segments et chaque segment est un ensemble de pages. On utilise une table de segments et une table de pages par segment pour retrouver une adresse.

Système multi-niveaux de pagination pure

Informations:

1) Une page de X kio veut dire : X mille octets donc X = 2**y bits

=> qu'il y a 2**y emplacement dans la page. Donc y bits pour encoder le déplacement.

2) De la, si on a le nombre de bits d'une adresse virtuelle (on va dire z bits)

=> z-y = Nbp (nombre de bits pour représenter le numéro de page)

3) Autre option,

=> taille mémoire / taille page = nombre de page

=> 2**Nbp = nombres de pages

=> l'adresse virtuelle sera les y + Nbp bits.

5) Le Nbp peut être divisé en n niveau de table de page équitable. (permet de charger une table seulement en mémoire pour retrouver une adresse en passant par les n niveaux)

6) Fonctionnement des n niveaux: Garder le premier niveau table et swap les autres tables de pages.

7) Convertir une adresse virtuelle en physique :mettre en binaire, prendre les Nbp bits et trouver les n entrés dans les tables de pages. On passe aux travers les n tables de pages. On prend l'adresse finale et on ajoute le déplacement y.

Trouver à partir d'un octet de l'espace d'adressage un numéro et déplacement:

=> numéro de page = octet / taille de page

=> déplacement = octet % taille de page

Prévention des interblocages

Solutions (dans le cas 3) :

- Exclusion mutuelle : Impossible de l'empêcher

- Détention et attente Chaque processus demande à la fois toutes les ressources dont il a besoin. (peut causer des famines et dur a mettre en place)

- Pas de réquisition : Sauvegarder l'état des ressources pour pouvoir interrompre un process (réquisitionner une ressource) et le reprendre plus tard.

- Attente circulaire : ordonner la séquence d'utilisation des ressources pour éviter les deadlocks

Allocation de mémoire

- premier ajustement **(First-fit)** => premier trou

- meilleur ajustement **(Best-fit)** => plus petit trou pouvant accueillir le bloc

- pire ajustement **(Worst-fit)** => plus grand trou pouvant accueillir le bloc

- par subdivision **(Buddy system)** => diviser la mémoire en puissance de deux jusqu'à ce qu'on fit le bloc. quand deux zones de même taille sont libres, elles sont fusionnées.

Politique de remplacement

Terminologie:

MMU (Memory Management Unit) : unité matérielle gérant les pages

TLB (Translation Lookaside Buffer) : cache gardant les dernières infos de la table des pages pour les traductions récentes

Politique de remplacement:

FIFO : first in first out (remplace la première page entré)

LRU : Least recently used (remplace la page ayant été utilisé il y a le plus longtemps)

Algorithme de l'horloge: est une approximation du LRU.

En gros, on fait un disque. Lorsqu'on place une donnée dans le disque, on met le bits à 1. Lorsqu'on veut remplacer une donnée, on remplace le premier bits trouvé qui est à 0 MAIS tous les bits à 1 sur lesquels ont passe sont mis à 0.

Algorithme de Belady

Utilité: Algorithme permettant de théorisé la politique de remplacement idéale et ainsi servir de modelé de comparaison. (impossible à mettre en place)

Fonctionnement: Quand on à un défaut de page, on choisit de remplacer la donné parmi nos choix actuels qui sera utilisé dans le plus longtemps dans le futur.

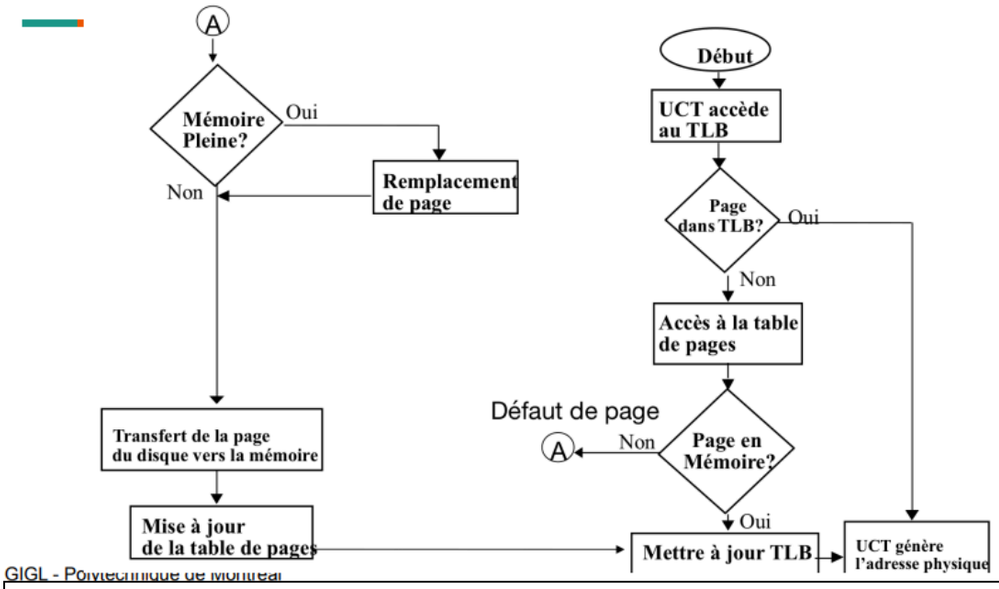
Noyau temps réels

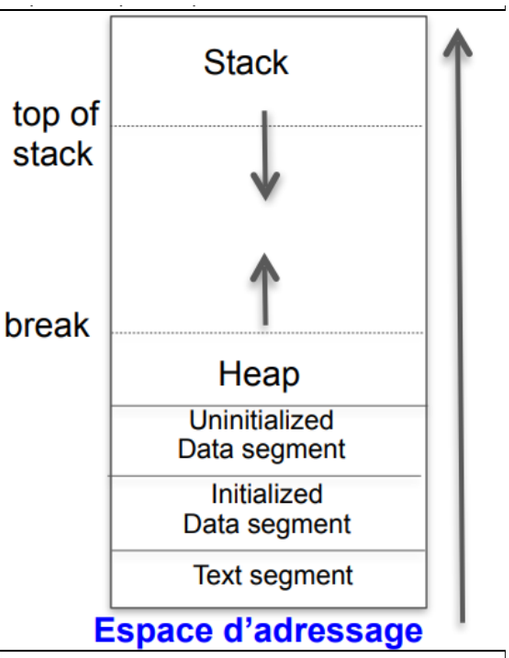
Définition: Un noyau temps réels est un noyau conçu pour réagir de façon très rapide et déterministe à des taches critiques. Les délais d'exécutions des taches sont donc bornés et prévisibles.

Tâche périodique : Le système doit garantir que la tâche est exécutée à intervalles réguliers sans interruption excessive pour respecter les échéances de la tâche.

Tâche apériodique :Le système doit être capable de détecter et de répondre rapidement à l'arrivée d'une tâche apériodique, en la traitant aussi efficacement que possible sans perturber le fonctionnement des autres tâches périodiques ou apériodiques en cours.

Pagination pure - Translation d'adresses





Espace d'adressage

Ordonnanceur

Définition: Partie du système d'exploitation (SE) qui se charge de gérer l'allocation du (des) processeur(s) aux processus/threads prêts.

Critères d'évaluation:

1) Taux d'utilisation d'un processeur (taux d'occupation).

2) Capacité de traitement d'un processeur : nombre de processus traités par unité de temps.

3) Temps de séjour d'un processus (temps de rotation ou de virement) : temps entre son admission et sa terminaison.

4) Temps de réponse d'un processus : temps entre son admission et le début de son exécution.

5) Temps d'attente d'un processus : somme de ses temps passés à l'état prêt.

6) Temps moyen de séjour, temps moyen de réponse et temps moyen d'attente

Type de politiques d'ordonnancement

Non préemptifs: Un processus ne va s'arrêter que s'il le décide ou que sa tache est terminée.

Type de politique :

– Premier arrivé, Premier servi (FCFS appelé aussi FIFO)

– Plus prioritaire d'abord, etc.

Préemptifs: L'os peut décider d'interrompre un processus bloquant pour gérer une tache plus urgente.

Type de politique :

- **circulaire**
(tourniquet/round robin) : liste fifo ou chaque processus est exécuté pendant un quantum de temps. La valeur du quantum est arbitrairement 10 à 100 fois le temps pour changer de processus.

- **files multiples** : des files fifo avec des priorités différentes auxquels sont associés des quantum de temps.

Problèmes:
peut entrainer de l'inversion de priorité => des processus moins importants bloques des processus importants, un processus ne peut jamais s'exécuter à cause d'une priorité trop faible, file bloqué par l'attente active d'une ressource.

Solutions: associer une priorité dynamique aux processus. Un processus qui consomme sont quantum de temps voit sa priorité réduite, alors qu'un processus en attente voit sa priorité augmenter.

Files et processeurs

Contexte:
Dans un système à multiprocesseur, le noyau doit être en mesure de répartir les processeurs à exécuter sur les multiples processus. Il faut donc stocker les processeurs dans une structure de donné que les processeurs (qui ont chacun une cache) pourront accéder.

Il y a deux stratégies possibles:

1) Une file pour tous les processeurs

Avantages: répartition équitable du temps CPU

Défauts:

- Doit assurer un accès en exclusion mutuelle à la file d'attente commune

- Plus faible localité au niveau des caches (plus de « cache miss »).

2) Une file par processeurs et une globale

Avantages: scalable et meilleur localité sur les caches

Défauts: risque de charge déséquilibrée entre les processeurs

Ordonnancement TR

Formules de condition pour déterminer si un ensemble est ordonnançable sans diagramme de Gantt:

RMA: $\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$

DMA: $\sum_{i=1}^{n} \frac{Ci}{Di} \leq n(2^{\frac{1}{n}} - 1)$

EDF: $\sum_{i=1}^{n} \frac{Ci}{Pi} \leq 1$ CS : $\sum_{i=1}^{n} \frac{Ci}{Di} \leq 1$

Traitement inversion priorité:

PIP, OCPP, ICPP

(héritage prio quand bloqué, max prio en réception ressources, max prio en demande de ressource)

Exercice 4 – Gestion de la mémoire

H. Boucheneb

On considère un système de pagination à 3 niveaux dans lequel les adresses virtuelles et physiques sont codées sur 48 bits. La taille d'une page est de 4 KiO. La taille de chaque table de pages, peu importe son niveau, est égale à 32 KiO. Chaque entrée d'une table de pages est composée de 8 octets.

1- Donnez le format de l'adresse virtuelle utilisée pour convertir une adresse virtuelle en une adresse physique. Expliquez comment vérifier si la page de l'adresse virtuelle 0x BA9 876 543 210 est présente en mémoire.

Donnez le format de l'adresse virtuelle :
L'adresse virtuelle est composée de 4 champs (3 niveaux de tables de pages + déplacement dans la page).
Le nombre de bits pour le déplacement dans la page est donné par la taille d'une page (4KiO = 2^{12}octets) => 12 bits.
La taille d'une table de page est 32KiO = 2^{15}octets. Chaque entrée d'une table de pages est composée de 8octets = 2^3octets. Le nombre d'entrées dans chaque table de pages est : $2^{15}/2^3 = 2^{12}$ entrées.
Le format est donc 12 bits (1er niveau) + 12 bits (2ième niveau) + 12 bits (3ième niveau) + 12 bits (déplacement).

Comment vérifier si la page de l'adresse virtuelle 0xBA9 876 543 210 est présente en mémoire ?
Les 12 bits de poids fort (c-à-d BA9) vont permettre de sélectionner une entrée dans la table du 1er niveau. Cette entrée pointe vers le début d'une table de pages de 2ième niveau. Les 12 bits suivants de l'adresse (c-à-d 876) vont permettre de sélectionner une entrée dans cette table de 2ième niveau. Cette entrée pointe, à son tour, vers le début d'une table de 3ième niveau. Les 12 bits suivants de l'adresse (c-à-d 543) vont permettre de sélectionner une entrée dans cette table de 3ième niveau. Cette entrée contient toutes les informations concernant la page référencée. Si le bit de présence est en mémoire, la page contenant l'octet référencé est en mémoire. Sinon, elle n'est pas en mémoire.

Dans un tel système, un processus A en cours d'exécution crée un processus fils (désigné dans ce qui suit par B). La création du processus B est réalisée par duplication selon le principe "copy-on- write". Au moment de la création de B, seules les pages 1, 5 et 7 de A sont en mémoire. Elles ont été respectivement chargées dans l'ordre 1, 5 et 7 dans les cadres 0, 2 et 3.

La partie de la mémoire physique réservée aux processus A et B est composée des cadres 0, 1, 2 et 3. Le cadre 1 est libre. Après la création du processus B, le processeur référence, dans l'ordre, les pages suivantes :
(7BW)(1BR)(5BR)(5BW)(1AW)(6BW)(6AR)(3BR)(3AR).
Où chaque triplet (X P W/R) signifie que le processeur référence la page X du processus P en mode écriture (W) ou en mode lecture (R).

Lorsqu'un défaut de page se produit et qu'un retrait de page est nécessaire, le système choisit une page parmi celles qui sont réservées aux processus A et B. L'algorithme de remplacement de pages utilisé est FIFO (First In First Out). Lorsqu'une page est chargée ou retirée de la mémoire, les tables de pages des processus qui partagent cette page sont mises à jour.

2- Donnez, en complétant le tableau suivant, l'évolution de l'état de la mémoire (la partie réservée aux processus A et B). Indiquez pour chaque page en mémoire les processus qui la partagent. Donnez le nombre de défauts de page provoqués par chacun des processus.

Les pages 1, 5 et 7 chargées dans l'ordre dans les cadres 0, 2 et 3. **Algorithme FIFO**

Cadre		7BW	1BR	5BR	5BW	1AW	6BR	6AR	3BR	3AR
0	1 AB	1 AB	1 AB	1 AB	5 B	5 B	5 B	5 B	5 B	5 B
1		7 B	7 B	7 B	7 B	7 B	7 B	7 B	3 AB	3 AB
2	5 AB	5 AB	5 AB	5 AB	5 A	1 A	1 A	1 A	1 A	1 A
3	7 AB	7 A	7 A	7 A	7 A	7 A	6 AB	6 AB	6 AB	6 AB

Le nombre de défauts de page provoqués par A est 1.
Le nombre de défauts de page provoqués par B est 4.

Exercice 5 – Ordonnancement temps réel Solution

H. Boucheneb

Tâche	Date d'arrivée (Oi)	Durée d'exécution (Ci)	Période=Échéance (Pi=Di)
T1	0	2 (ER)	6
T2	0	2 (EE)	8
T3	0	5 (RRRRR)	12

Prio(T1)>Prio(T2) > Prio(T3)

L'intervalle de simulation [0,PPCM(6,8,12)]

1. Ces tâches sont-elles ordonnançables selon RMA ? Est-ce qu'il y a une inversion de priorités ? Justifiez votre réponse.

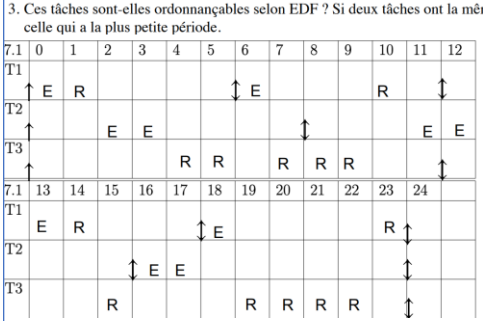
Ces tâches ne sont pas ordonnançables selon RM car la tâche T1 va manquer son échéance à la da 12 (date de sa troisième activation). Elle n'aura fini son traitement avant sa troisième activation.

Oui, il y a une inversion de priorités entre T1 et T3 à la date 7 et durera jusqu'à ce que T3 libère la ressource R. Durant tout ce temps, l'exécution de la tâche prioritaire T1 est bloquée par la tâche T3 moins prioritaire.

2. Ces tâches sont-elles ordonnançables selon RMA, dans le cas où le protocole PIP est utilisé pour traiter les inversions de

Ces tâches sont ordonnançables selon RMA dans le cas où le protocole PIP est utilisé pour traiter les inversions de priorités.

3. Ces tâches sont-elles ordonnançables selon EDF ? Si deux tâches ont la même priorité dynamique, l'ordonnanceur va élire celle qui a la plus petite période.



Ces tâches sont ordonnançables selon EDF car toutes les tâches respectent leurs échéances.

Ordonnancement de processus

H. Boucheneb

Comment gérer l'allocation de processeur(s) aux processus/threads prêts ?

Ordonnancements non préemptifs

- **Ordonnancement FIFO ou à priorités :**
  - Le système d'exploitation choisit le processus en tête de file (FIFO) ou le plus prioritaire (à priorités).
  - Il lui alloue le processeur jusqu'à ce qu'il se termine, se bloque ou cède le processeur. Il n'y a pas de réquisition.

Ordonnancements préemptifs

- **Ordonnancement circulaire (tourniquet) :**
  - Le système d'exploitation choisit le processus en tête de file (FIFO).
  - Il lui alloue le processeur jusqu'à ce qu'il se termine, se bloque, cède le processeur ou ait consommé son quantum.
  - Les nouveaux processus et les processus qui basculent vers l'état prêt sont toujours insérés à la fin de la file d'attente.

➔ performances sensibles à la valeur du quantum (temps d'attente moyen et nombre de commutations de contexte).

Ordonnancement de processus (2)

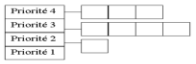
H. Boucheneb

Ordonnancements préemptifs (suite)

- **Ordonnancement à priorités :**
  - Le système d'exploitation choisit le processus le plus prioritaire.
  - Le processus élu est exécuté jusqu'à ce qu'il se termine, se bloque, cède le processeur ou un processus plus prioritaire arrive ou devient prêt.

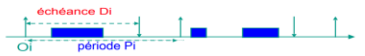
➔ problèmes de famine et d'inversions de priorités
➔ solutions basées sur des priorités dynamiques (non évidentes).

- **Ordonnancement à files multiples** (combinant plusieurs politiques d'ordonnancement) :
  - Le processus élu est celui en tête de la file d'attente la plus prioritaire.
  - Il est exécuté jusqu'à ce qu'il se termine, se bloque, cède le processeur, ou un processus plus prioritaire arrive ou devient prêt.
  - Il pourrait être aussi suspendu, s'il a consommé son quantum, dans le cas où l'ordonnancement appliqué à son niveau de priorité est circulaire.
  - Dans le cas où les priorités sont dynamiques, la priorité dynamique d'un processus est généralement recalculée lorsqu'il passe à l'état bloqué ou à l'état prêt.



Ordonnancement temps réel (préemptif à priorités)

- Soit un ensemble de n tâches T1,…, Tn périodiques indépendantes. Chaque tâche Ti a :
  - une période Pi,
  - un temps d'exécution du traitement périodique Ci,
  - une date de première activation Oi et
  - éventuellement une échéance Di avec Di≤Pi.



- **Les tâches** T1,…, Tn sont ordonnançables selon une politique d'ordonnancement, si à chaque fois qu'une tâche est activée (passe à l'état prêt), elle va pouvoir terminer son traitement périodique avant sa prochaine activation (ou son échéance si Di <Pi).

Si les tâches **T1, …, Tn** sont ordonnançables dans l'intervalle de temps **[0, Max(O1,…,On)+PPCM(P1,…,Pn)]** alors elles sont ordonnançables (dans [0,∞[).

PPCM= Plus Petit Multiple Commun

Ordonnancement temps réel (préemptif à priorités) (2)

- Soit un ensemble de n tâches T1,…Tn **périodiques indépendantes**.

Rate monotonic priority assignment (RMA)
- Priorités inversement proportionnelles à leurs périodes ➔ Priorités statiques.
- Di = Pi pour i=1,n.
- Ordonnançable si (condition suffisante de *Liu et Layland*) : $\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n}-1)$

n	Utilization Bound
1	100.0%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%

Deadline monotonic priority assignment (DMA)
- Priorités inversement proportionnelles à leurs *deadlines* ➔ *Priorités statiques*
- *Ordonnançable si :* $\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}}-1)$

Earliest Deadline First (EDF)
- La tâche la plus prioritaire est celle dont l'échéance absolue est la plus proche ➔ Priorités dynamiques
- Il est applicable aussi bien pour des tâches périodiques qu'apériodiques.

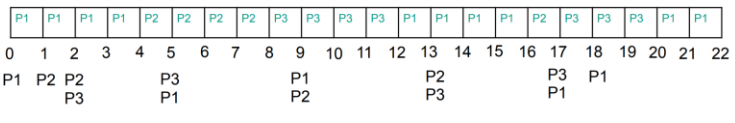
- **Si Pi=Di pour i=1,n ➔ Ordonnançable ssi** $\sum_{i=1}^{n} \frac{C_i}{D_i} \leq 1$
- **Si Pi≥Di pour i=1,n ➔ Ordonnançable si** $\sum_{i=1}^{n} \frac{C_i}{D_i} \leq 1$

Tâches dépendantes
=> Inversion de priorités (protocole PIP)

Supposez trois processus P1, P2 et P3 avec les caractéristiques suivantes :

Processus	Date d'arrivée (Oi)	Durée d'exécution (Ci)
P1	0	10
P2	1	5
P3	2	7

Donnez le digramme de Gant de l'exécution de ces processus dans le cas d'un ordonnancement circulaire de quantum de 4. Donnez les temps de séjour et d'attente moyens (TMS et TMA). Les temps de commutation de contexte sont supposés nuls.



TMA = [((12-4) + (20-16)) + ((4-1)+(16-8)) + ((8-2)+(17-12))] / 3 = 11.33
TMS = (22 + (17-1) + (20-2)) / 3 = 18.66

Exercice 8 : Ordonnancement

Processus	Date d'arrivée (Oi)	Durée d'exécution (Ci)
P1	0	10 (3, 6 en section critique, 1)
P2	1	5 (2, 2 en section critique, 1)
P3	2	7

Donner le digramme de Gant de l'exécution des processus P1, P2 et P3, pour chacun des cas suivants :

a) une attente passive d'accès aux sections critiques. Les sections critiques sont encadrées par P(mutex) et V(mutex)