Ordonnancement
Systèmes Temps-Réels

Plan

- Caractéristiques systèmes temps-réels
- Systèmes périodiques
- Systèmes sporadiques
- Gestion ressources autre que CPU

Synchrone / Asynchrone

- Deux types de systèmes
  - Synchrone
    - Existence d'une base de temps commune,
    - Les évènements n'arrivent pas n'importe quand
  - Asynchrone
    - Pas d'hypothèse sur les instants où les évènements peuvent se produire
- Le monde synchrone est plus « simple », le monde réel est plutôt asynchrone

Déterminisme

- Pouvoir garantir que le système respectera ses spécifications, notamment temporelles, pendant sa durée de vie
  - Ré-exécution donne des résultats identiques
- Méthodologie
  - Déterminer les cas pires
  - Conditions de faisabilité (CF)
  - Déterminer valeurs numériques CF
  - Vérifier

Algorithme déterministe

- Temps maximum d'exécution garanti
- Indépendant du contexte courant
- Indépendant de la valeur des arguments
- Principe valable pour toutes les fonctions d'un même service
  - allocation / libération pour gestion mémoire
- S'applique aux séquences d'exclusion mutuelle
  - Pas d'effet(s) de bord sur reste du système

Caractéristiques temporelles

- Durée maximum (pire cas) d'une thread τ_i : C_i
  - Thread seule sans interruption
  - Par analyse ou par mesure
- (Pire) temps de réponse d'une thread τ_i : R_i
  - Temps entre demande activation et réponse
  - Prend en compte délai dans exécution induit par les autres threads et l'overhead OS
- $R_i \geq C_i$

Contraintes Temporelles

- Échéance de terminaison au plus tard
- Thread τ_i activée à instant t_i doit être terminée au plus tard à instant $t_i + D_i$
- D_i : échéance relative
- $t_i + D_i$: échéance absolue

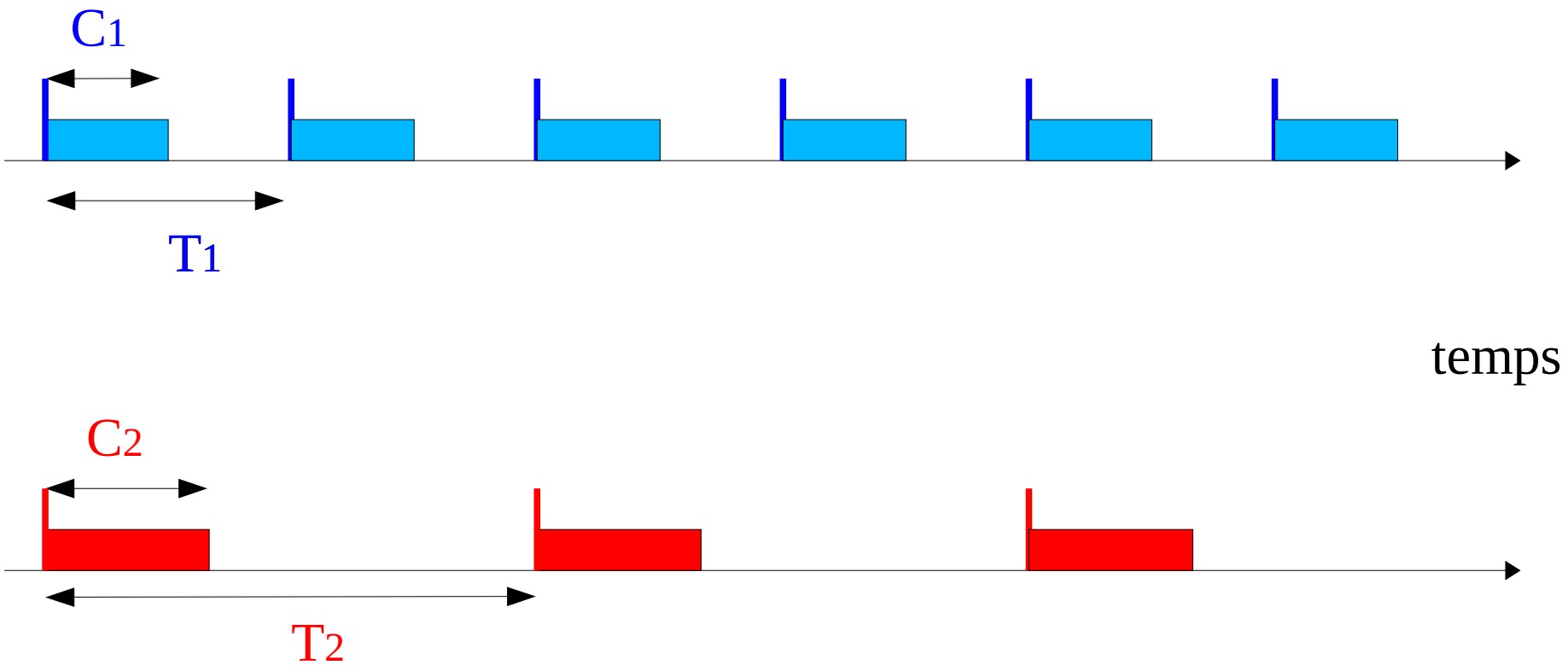
Plan

- Caractéristiques systèmes temps-réels
- Systèmes périodiques
- Systèmes sporadiques
- Gestion ressources autre que CPU

Systèmes Périodiques

- Chaque thread τ_i activée périodiquement
  - Période activation T_i
  - Échéance relative D_i (en général $T_i = D_i$)
- La $k^{ième}$ instance de la thread τ_i
  - Est activée à $(k-1)\ T_i$
  - Doit être terminée au plus tard à $k\ T_i$
- Hyper période = PPCM(T_i) *i = 1,…,n*

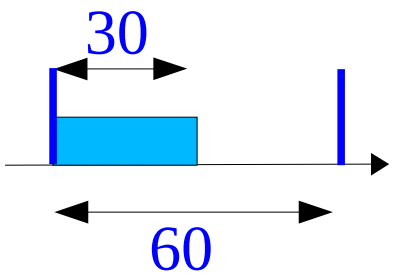
Systèmes Périodiques

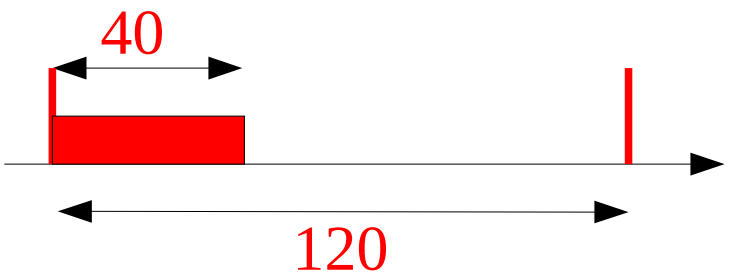


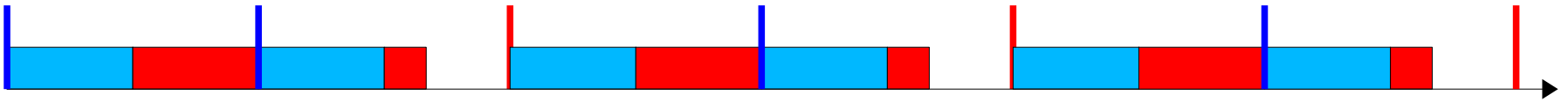
Ordonnancement Rate Monotonic

- Priorité déterminée en fonction de la période
- Plus la période est petite, plus la priorité est élevée
- Optimal
  - pour systèmes périodiques
  - Avec ordonnancement préemptif
  - $\forall$ i = 1,.. n $D_i = T_i$

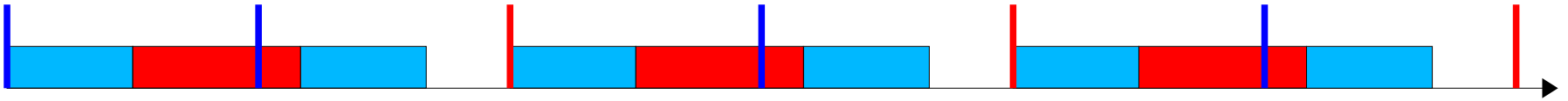
Ordonnancement Rate Monotonic







Ordonnancement "optimal" (minimise Nb. changement de contextes)

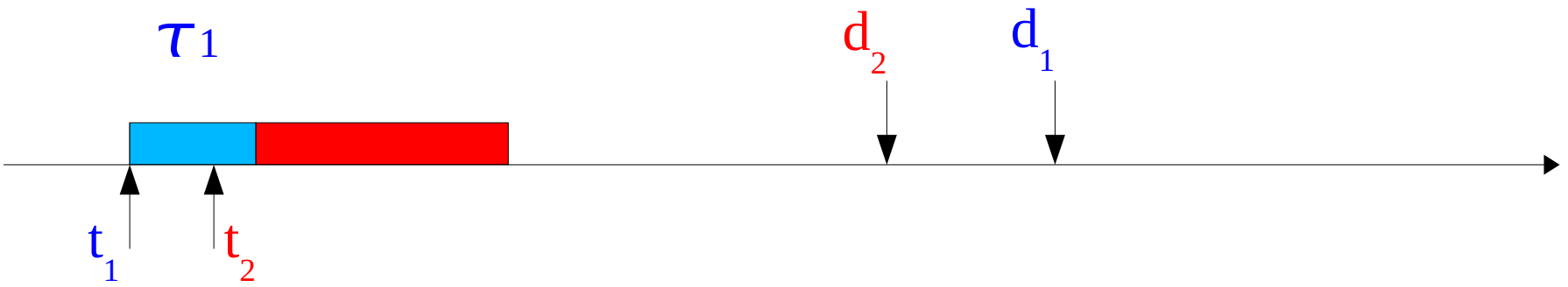


Plan

- Caractéristiques systèmes temps-réels
- Systèmes périodiques
- Systèmes sporadiques
- Gestion ressources autre que CPU

Systèmes Sporadiques

- Chaque thread τ_i
  - Activée à un temps t_i
  - Durée maximum d'exécution C_i
  - Échéance absolue $d_i = t_i + D_i$
- Ordonnancement possible



Systèmes Sporadiques

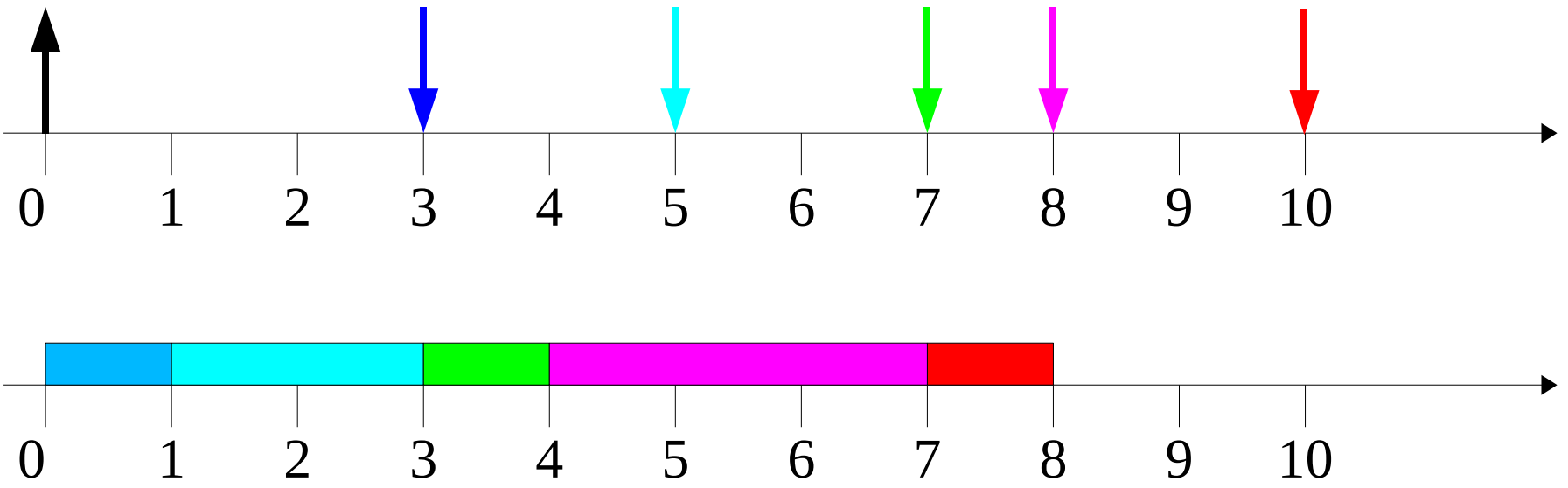
- Ordonnancement statique
  - Ensemble (t_i ,c_i ,d_i) i = 1,...,n connu avant exécution
  - Construire un ordonnancement qui respecte les échéances de chaque thread (*ordonnancement faisable*)
- Ordonnancement dynamique
  - A chaque "moment d'ordonnancement", déterminer la prochaine thread à exécuter
  - Satisfaire les échéances de toutes les threads

Ordonnancement **EDF**

- **E**arliest **D**eadline **F**irst
- Ordonnanceur dynamique
- Donne priorité à la thread la plus proche de son échéance de terminaison
- Doit trier les threads en fonction de leurs échéances
- Optimal si système non surchargé
- Comportement non-prédictible en cas de surcharge

EDF / même temps d'activation

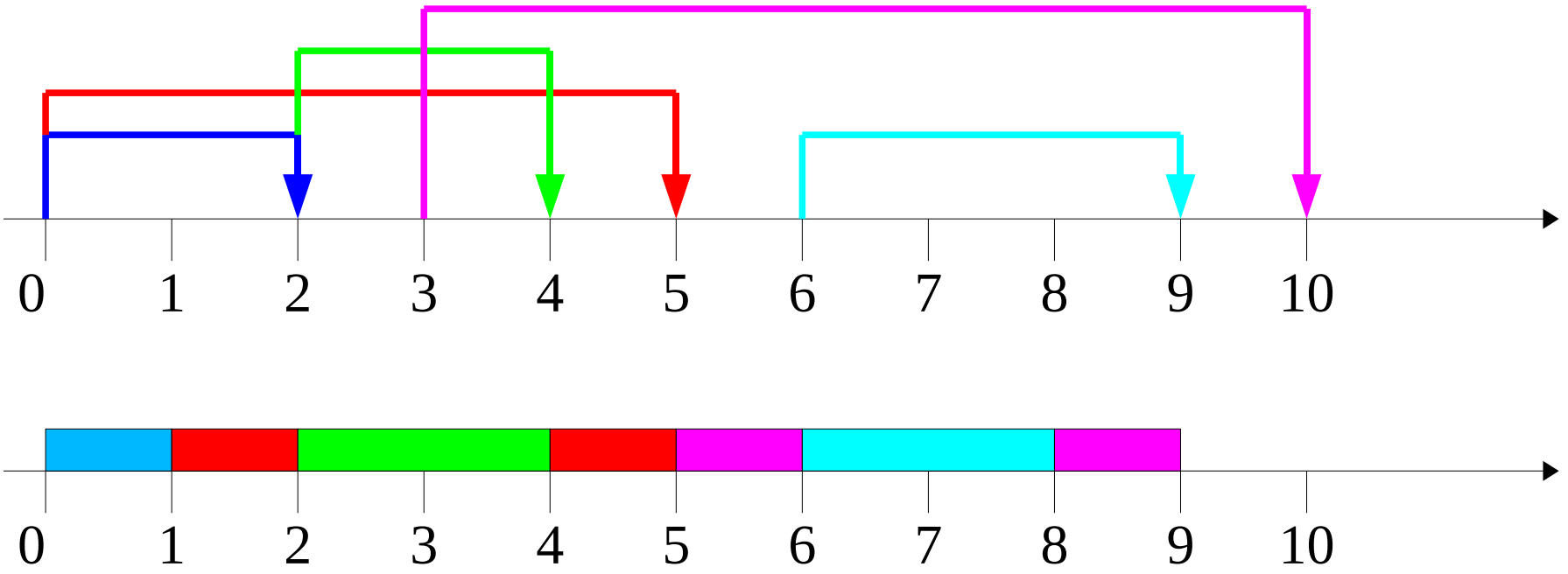
5 threads : (0,1,3), (0,1,10), (0,1,7), (0,3,8), (0,2,5)



préemption pas nécessaire

EDF / temps d'activation différents

5 threads : (0,1,2), (0,2,5), (2,2,4), (3,2,10), (6,2,9)



=> Préemption nécessaire pour satisfaire échéance de (2,2,4)

Inconvénients (1)

- Ordonnancements basés sur le temps
  - Supposent CPU est la seule ressource partagée
- Dépendants
  - Caractéristiques matériel
    - CPU (instructions, fréquence, cache(s), etc...)
    - Performances bus (mémoire, I/O, etc)
  - Compilateurs
- Très sensibles aux évolutions du logiciel
  - Correction de bugs
  - Ajout de tâches

Inconvénients (2)

- Décomposition des activités en blocs d'exécution synchrones :
  thread = (activation, durée maximum, échéance)
  - Figée
  - [relativement] simple
- Décomposition en étapes asynchrones de priorités différentes :
  étape = interruption / thread
  - Plus souple
  - Plus complexe

Plan

- Caractéristiques systèmes temps-réels
- Systèmes périodiques
- Systèmes sporadiques
- Gestion ressources autre que CPU

Gestion Ressources non CPU

- Pour toute ressource "non réquisitionnable"
- Ordonnancement des threads en attente
  - En FIFO
  - Fonction de leur priorité
  - Fonction des ressources déjà acquises
- Optimiser délai d'occupation ressources
  - augmente temporairement priorité threads propriétaires

Interblocage Allocation Ressource

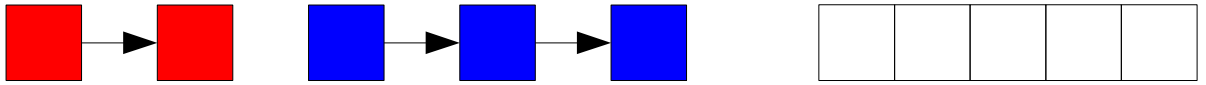
- Pool de ressources allouées individuellement
- Quand pool vide, bloque thread dans file d'attente
- 10 ressources libres dans pool
- Thread basse priorité : besoin de 3
- Thread priorité intermédiaire : besoin de 6
- Thread haute priorité : besoin de 7

Interblocage Allocation Ressource

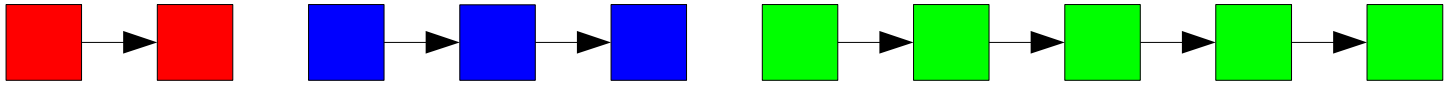
10 ressources libres dans pool

1. thread basse acquiert 2 ressources

2. thread intermédiaire préempte thread basse et acquiert 3 ressources

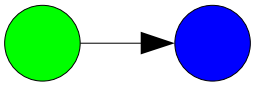


3. thread haute préempte thread intermédiaire et acquiert 5 ressources

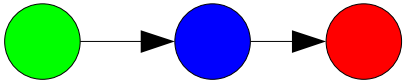


4. thread haute priorité mise en attente

5. thread priorité intermédiaire mise en attente



6. thread basse priorité mise en attente



Solutions Interblocage

- Ressources "anonymes"
- Allocation en 2 étapes
  1) Réserver nombre de ressources
     bloquer si NB disponibles < NB demandées
  2) Prendre ressources réservées une par une
- Solution plus fluide, si applicable
  - Bloquer uniquement pour première ressource
  - S'adapter à ce qui est disponible
    => effectuer plusieurs opérations si nécessaire