Systèmes d'exploitations

Cours 4: Interruption et ordonnancement

Matthieu Lemerre
CEA LIST

Année 2023-2024

Sommaire

1. Cours 4: Interruptions; politiques d'ordonnancement
   - Interruptions
   - Ordonnancement
   - Conclusion

But du cours

Comprendre:

- Comment fonctionne le multi-threading préemptif?
- Comment traiter efficacement des évènements sporadiques?
- Quels sont les choix possibles de politique d'ordonnancement?

But du cours

Comprendre:

- Comment fonctionne le multi-threading préemptif?
- Comment traiter efficacement des évènements sporadiques?
- Quels sont les choix possibles de politique d'ordonnancement?

Ce cours vous sera utile si vous devez:

- Écrire un pilote de périphérique

But du cours

Comprendre:

- Comment fonctionne le multi-threading préemptif?
- Comment traiter efficacement des évènements sporadiques?
- Quels sont les choix possibles de politique d'ordonnancement?

Ce cours vous sera utile si vous devez:

- Écrire un pilote de périphérique
- Concevoir ou implanter un système embarqué (objet connecté, contrôle-commande...)

But du cours

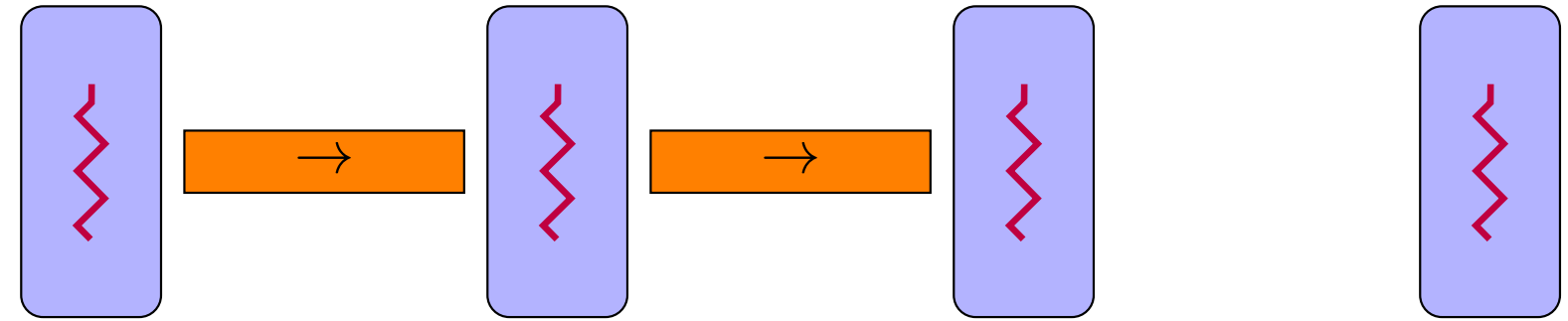
Comprendre:

- Comment fonctionne le multi-threading préemptif?
- Comment traiter efficacement des évènements sporadiques?
- Quels sont les choix possibles de politique d'ordonnancement?

Ce cours vous sera utile si vous devez:

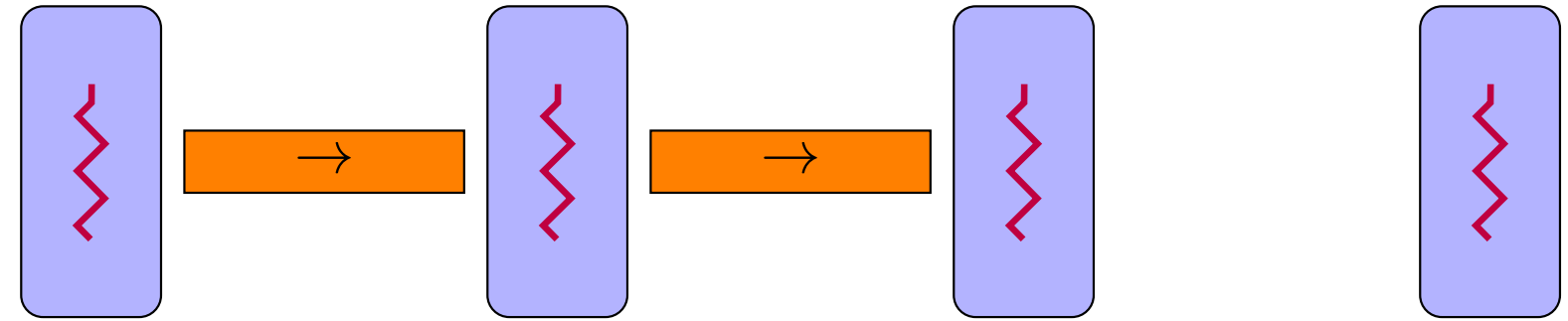
- Écrire un pilote de périphérique
- Concevoir ou implanter un système embarqué (objet connecté, contrôle-commande...)
- Optimiser un système à haute performance
  - trading haute fréquence
  - contrôle-commande de moteur diesel
  - équipement scientifique

Le polling



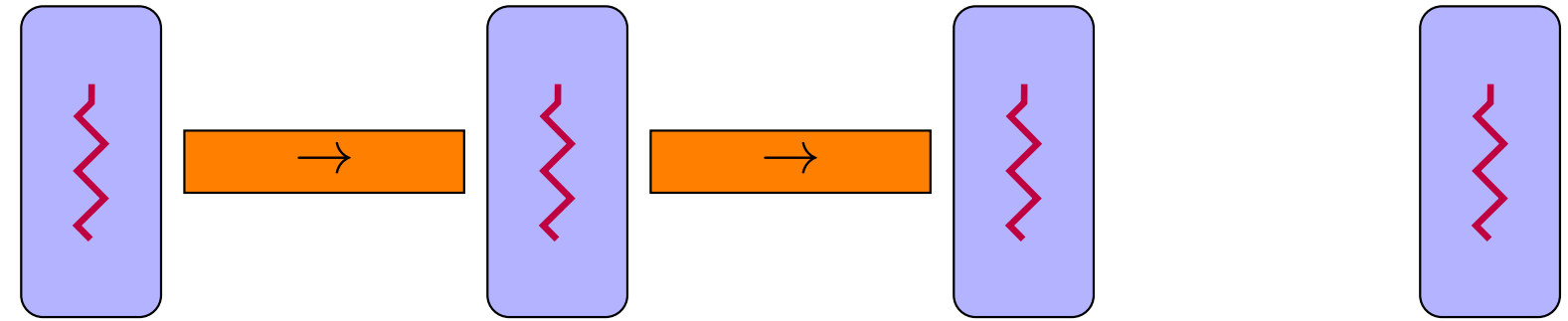
- Pour implanter l'atelier télégraphique 2.0 on peut:
  - Exécuter les threads "entrée clavier" et "sortie led" suffisament souvent
  - À chaque exécution, interroger le clavier et l'horloge pour voir si une touche a été appuyée ou si il est l'heure d'allumer la LED.
- Cela s'appelle le *polling*.

Le polling



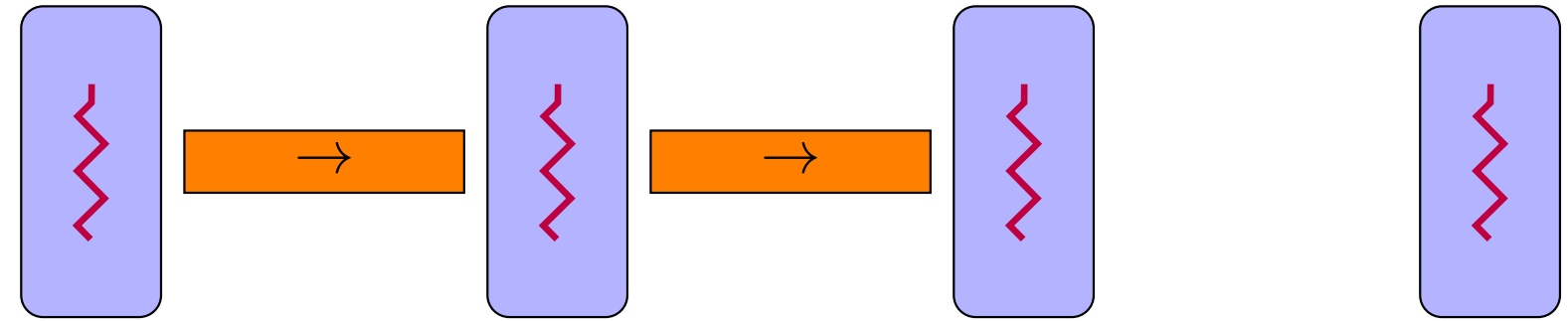
- Pour implanter l'atelier télégraphique 2.0 on peut:
  - Exécuter les threads "entrée clavier" et "sortie led" suffisament souvent
  - À chaque exécution, interroger le clavier et l'horloge pour voir si une touche a été appuyée ou si il est l'heure d'allumer la LED.
- Cela s'appelle le *polling*.
- Difficultés:

Le polling



- Pour implanter l'atelier télégraphique 2.0 on peut:
  - Exécuter les threads "entrée clavier" et "sortie led" suffisament souvent
  - À chaque exécution, interroger le clavier et l'horloge pour voir si une touche a été appuyée ou si il est l'heure d'allumer la LED.
- Cela s'appelle le *polling*.
- Difficultés:
  - Lorsque l'évènement n'est pas survenu, on a perdu du temps processeur
  - Si on ne vérifie pas assez souvent, on peut perdre des entrées
  - $\Rightarrow$ Il faut bien spécifier ce que signifie "suffisament souvent".

Le polling



- Pour implanter l'atelier télégraphique 2.0 on peut:
  - Exécuter les threads "entrée clavier" et "sortie led" suffisament souvent
  - À chaque exécution, interroger le clavier et l'horloge pour voir si une touche a été appuyée ou si il est l'heure d'allumer la LED.
- Cela s'appelle le *polling*.
- Difficultés:
  - Lorsque l'évènement n'est pas survenu, on a perdu du temps processeur
  - Si on ne vérifie pas assez souvent, on peut perdre des entrées
  - $\Rightarrow$ Il faut bien spécifier ce que signifie "suffisament souvent".
- Comment peut-on rendre cela plus efficace (si nécessaire)?

Les interruptions

Définition (Interruption)

- Mécanisme matériel permettant de signaler au processeur un évènement réquerant son attention immédiate.
- Le processeur répond à cet évènement en exécutant une fonction appellée service d'interruption (*interrupt handler*).

Les interruptions

Définition (Interruption)

- Mécanisme matériel permettant de signaler au processeur un évènement réquerant son attention immédiate.
- Le processeur répond à cet évènement en exécutant une fonction appellée service d'interruption (*interrupt handler*).

Les interruptions sont de deux types:

Interruptions matérielles provient d'un périphérique matériel

- Entrée clavier, arrivée d'un paquet réseau, etc.
- Fin de traitement d'une lecture ou écriture disque, d'un envoi de paquet réseau...
- Réveil à une heure programmée d'une horloge (*timer*)

Les interruptions

Définition (Interruption)

- Mécanisme matériel permettant de signaler au processeur un évènement réquerant son attention immédiate.
- Le processeur répond à cet évènement en exécutant une fonction appellée service d'interruption (*interrupt handler*).

Les interruptions sont de deux types:

Interruptions matérielles provient d'un périphérique matériel

- Entrée clavier, arrivée d'un paquet réseau, etc.
- Fin de traitement d'une lecture ou écriture disque, d'un envoi de paquet réseau...
- Réveil à une heure programmée d'une horloge (*timer*)

Interruptions logicielles (exceptions) provient du processeur lui-même

- Division par zéro
- Accès à une zone mémoire interdit ou impossible
- Interruptions volontaires (*trap*)

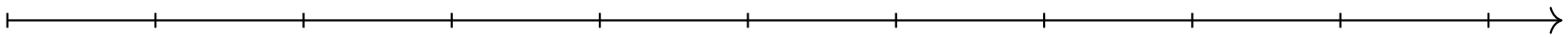
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

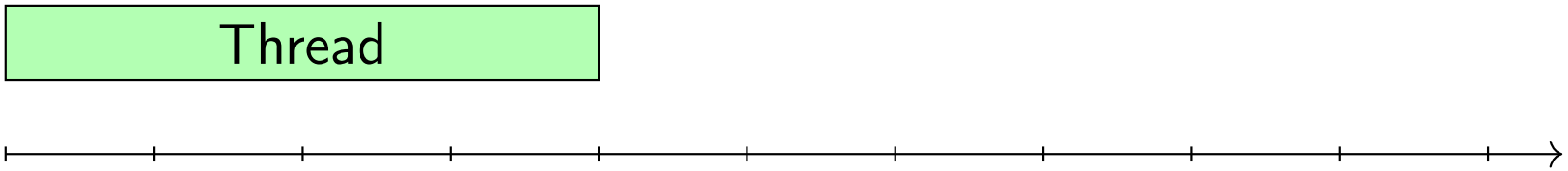
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

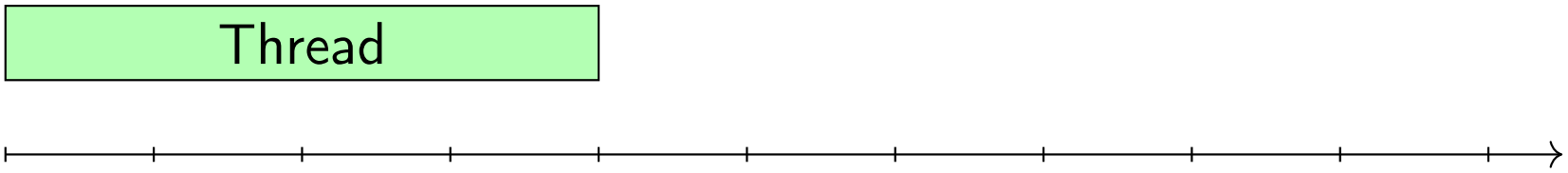
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

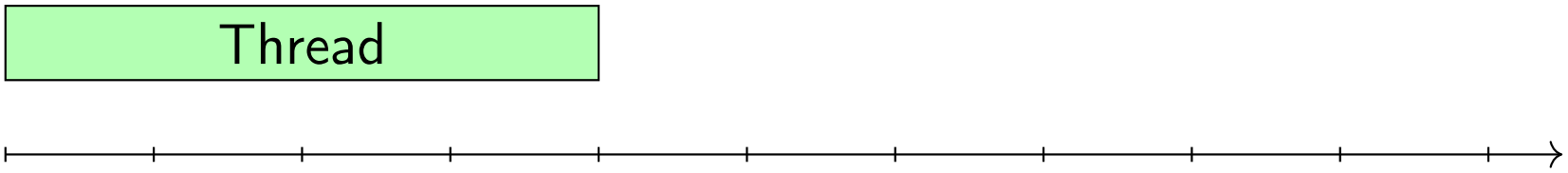
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

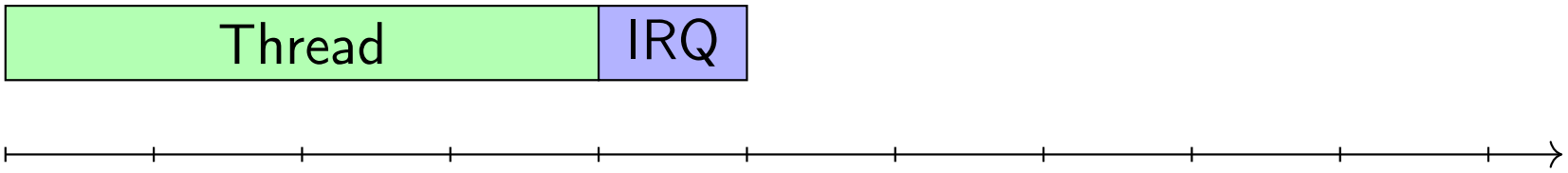
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c = get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

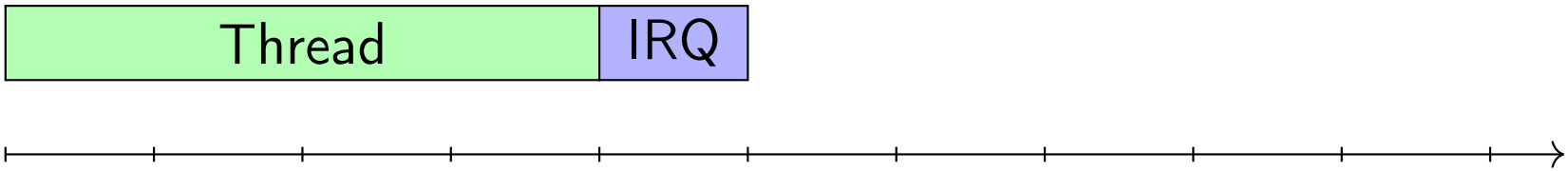
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

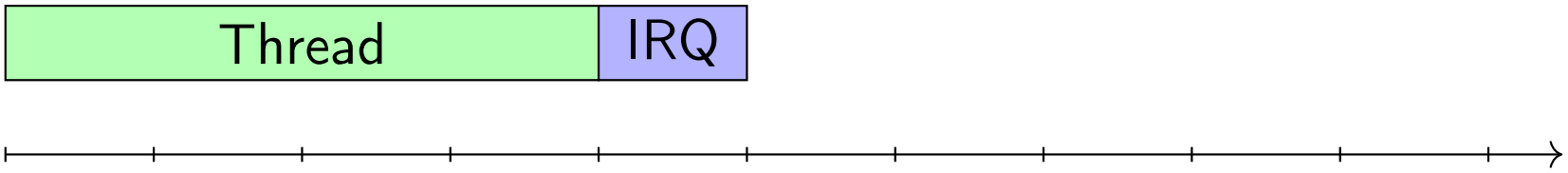
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

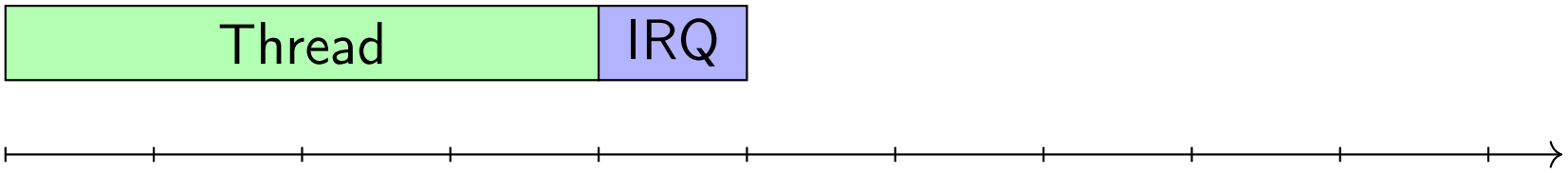
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

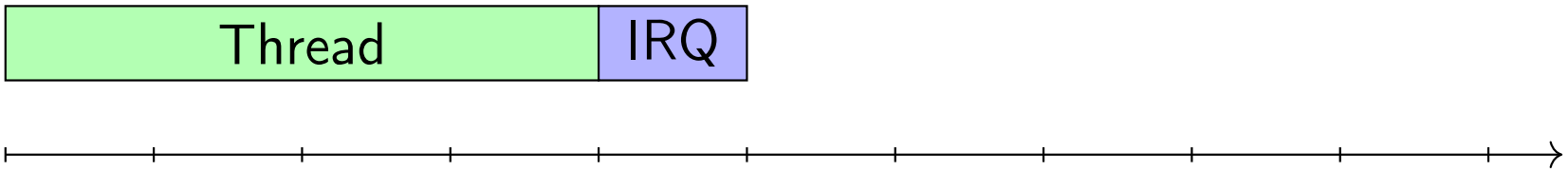
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

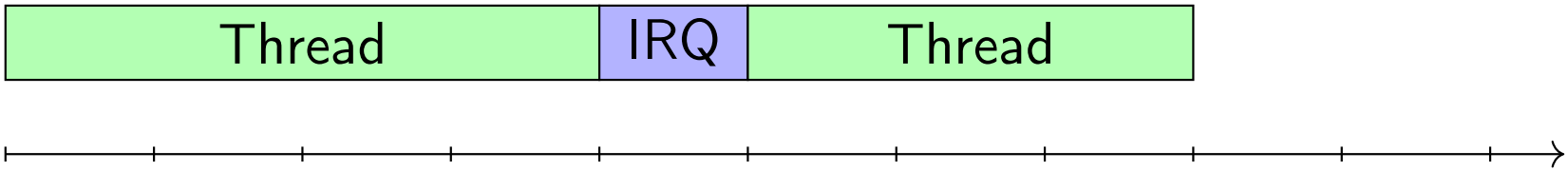
Fonctionnement des interruptions (1)

Exemple (Thread)

```
int a = 0;
int b = 20;
a++;
b++;
⋮
```

Exemple (Interrupt handler)

```
char c =
get_keyboard_value();
acknowledge_interrupt();
kbd_input[idx++] = c;
unblock_thread();
return;
```



- La routine d'interruption, généralement brève, interrompt le traitement du thread en cours.

Fonctionnement des interruptions (2)

Lors d'une interruption:

- pc $\leftarrow$ une adresse fixe (dépend du numéro d'interruption).
  - Qu'est-ce que le processeur doit faire d'autre?

Fonctionnement des interruptions (2)

Lors d'une interruption:

- `pc` $\leftarrow$ une adresse fixe (dépend du numéro d'interruption).
  - Sauvegarder l'ancien `pc`!
  - Souvent : sauvegarde automatique de `pc`, `sp`, `flags`, et d'autres registres
    - Sur pile, registres systèmes...
- `sp` $\leftarrow$ l'adresse d'une pile du noyau

Fonctionnement des interruptions (2)

Lors d'une interruption:

- `pc` $\leftarrow$ une adresse fixe (dépend du numéro d'interruption).
  - Sauvegarder l'ancien `pc`!
  - Souvent : sauvegarde automatique de `pc`, `sp`, `flags`, et d'autres registres
    - Sur pile, registres systèmes...
- `sp` $\leftarrow$ l'adresse d'une pile du noyau
- Exécution de la routine

Fonctionnement des interruptions (2)

Lors d'une interruption:

- `pc` $\leftarrow$ une adresse fixe (dépend du numéro d'interruption).
  - Sauvegarder l'ancien `pc`!
  - Souvent : sauvegarde automatique de `pc`, `sp`, `flags`, et d'autres registres
    - Sur pile, registres systèmes...
- `sp` $\leftarrow$ l'adresse d'une pile du noyau
- Exécution de la routine
- Restauration de tous les registres du thread interrompu

Fonctionnement des interruptions (2)

Lors d'une interruption:

- `pc` $\leftarrow$ une adresse fixe (dépend du numéro d'interruption).
  - Sauvegarder l'ancien `pc`!
  - Souvent : sauvegarde automatique de `pc`, `sp`, `flags`, et d'autres registres
    - Sur pile, registres systèmes...
- `sp` $\leftarrow$ l'adresse d'une pile du noyau
- Exécution de la routine
- Restauration de tous les registres du thread interrompu
- Restauration de `pc` (instruction `return from interrupt`)

Interruption pendant le traitement des interruptions

- Risque d'écraser les registres sauvegardés
- Quelle solution?

Interruption pendant le traitement des interruptions

- Risque d'écraser les registres sauvegardés
- Les nouvelles interruptions qui arrivent sont *masquées* (mises en attentes)
  - Instructions systèmes pour masquer ou démasquer les interruptions

Interruption pendant le traitement des interruptions

- Risque d'écraser les registres sauvegardés

- Les nouvelles interruptions qui arrivent sont *masquées* (mises en attentes)
  - Instructions systèmes pour masquer ou démasquer les interruptions
- Le masquage des interruptions introduit de la latence dans le traitement des interruptions

Interruption pendant le traitement des interruptions

- Risque d'écraser les registres sauvegardés

- Les nouvelles interruptions qui arrivent sont *masquées* (mises en attentes)
  - Instructions systèmes pour masquer ou démasquer les interruptions
- Le masquage des interruptions introduit de la latence dans le traitement des interruptions
  - $\rightarrow$ On préfère que les interrupt handlers soient courts.
  - Découpage en:
    - une partie rapide (le minimum à traiter, par ex. récupération d'une valeur)
    - une partie plus lente (qui peut être faite dans un thread plus tard)

Interruption pendant le traitement des interruptions

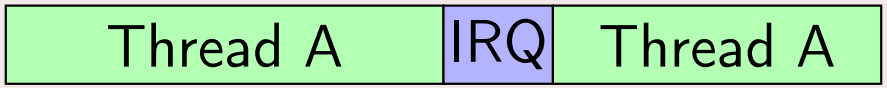
- Risque d'écraser les registres sauvegardés

- Les nouvelles interruptions qui arrivent sont *masquées* (mises en attentes)
  - Instructions systèmes pour masquer ou démasquer les interruptions
- Le masquage des interruptions introduit de la latence dans le traitement des interruptions
  - $\rightarrow$ On préfère que les interrupt handlers soient courts.
  - Découpage en:
    - une partie rapide (le minimum à traiter, par ex. récupération d'une valeur)
    - une partie plus lente (qui peut être faite dans un thread plus tard)
- Variantes: nested interrupts, non-masquable interrupts, interrupt priority

Interruption et ordonnancement

Interruption normale

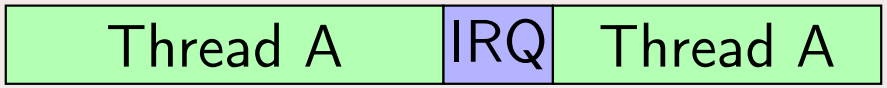
1. sauvegarde du contexte courant
2. exécution de l'interrupt handler
3. restauration du contexte courant



Interruption et ordonnancement

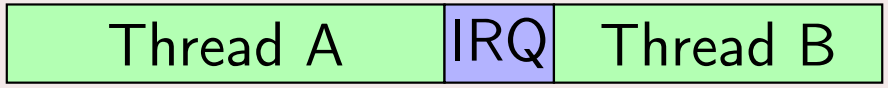
Interruption normale

1. sauvegarde du contexte courant
2. exécution de l'interrupt handler
3. restauration du contexte courant



Préemption

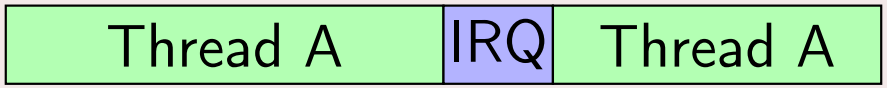
1. sauvegarde du contexte courant
2. exécution de l'interrupt handler
3. restauration d'un autre contexte



Interruption et ordonnancement

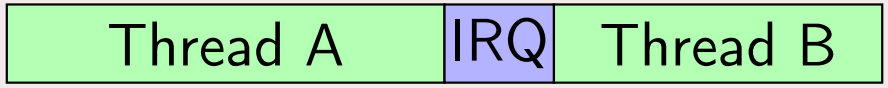
Interruption normale

1. sauvegarde du contexte courant
2. exécution de l'interrupt handler
3. restauration du contexte courant



Préemption

1. sauvegarde du contexte courant
2. exécution de l'interrupt handler
3. restauration d'un autre contexte



Interruption et préemption

Les interruptions sont le mécanisme matériel à la base de l'ordonnancement préemptif.

Sommaire

1. Cours 4: Interruptions; politiques d'ordonnancement
   - Interruptions
   - Ordonnancement
   - Conclusion

Définitions

Définition (Ordonnancement)

Attribution, au cours du temps, de *travaux* à des *resources*

- Notion temporelle: par ex. l'allocation statique de mémoire n'est pas un ordonnancement.
- Travaux: threads, pages à imprimer, paquets réseaux...
- Resources: CPU, imprimantes, liens réseaux...

Définitions

Définition (Ordonnancement)

Attribution, au cours du temps, de *travaux* à des *resources*

- Notion temporelle: par ex. l'allocation statique de mémoire n'est pas un ordonnancement.
- Travaux: threads, pages à imprimer, paquets réseaux...
- Resources: CPU, imprimantes, liens réseaux...

Définition (Politique d'ordonnancement)

Contraintes ou algorithme permettant de décider, à tout instant, quel travail exécuter parmi les travaux prêts.

Définitions

Définition (Ordonnancement)

Attribution, au cours du temps, de *travaux* à des *resources*

- Notion temporelle: par ex. l'allocation statique de mémoire n'est pas un ordonnancement.
- Travaux: threads, pages à imprimer, paquets réseaux...
- Resources: CPU, imprimantes, liens réseaux...

Définition (Politique d'ordonnancement)

Contraintes ou algorithme permettant de décider, à tout instant, quel travail exécuter parmi les travaux prêts.

Ordonnanceur

Code qui implémente une politique d'ordonnancement

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes
- Coût de *migration* (d'une ressource à une autre)

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes
- Coût de *migration* (d'une ressource à une autre)
  - Coût élevé: déménager une expérience d'une ligne de lumière au synchrotron soleil
  - Coût faible: changer de couteau

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes
- Coût de *migration* (d'une ressource à une autre)
  - Coût élevé: déménager une expérience d'une ligne de lumière au synchrotron soleil
  - Coût faible: changer de couteau
- Critère d'optimisation:

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes
- Coût de *migration* (d'une ressource à une autre)
  - Coût élevé: déménager une expérience d'une ligne de lumière au synchrotron soleil
  - Coût faible: changer de couteau
- Critère d'optimisation:

  Débit Maximiser le nombre de tâches faites

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes
- Coût de *migration* (d'une ressource à une autre)
  - Coût élevé: déménager une expérience d'une ligne de lumière au synchrotron soleil
  - Coût faible: changer de couteau
- Critère d'optimisation:

  Débit Maximiser le nombre de tâches faites

  Latence Minimiser les temps de réponse à des évènements

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes
- Coût de *migration* (d'une ressource à une autre)
  - Coût élevé: déménager une expérience d'une ligne de lumière au synchrotron soleil
  - Coût faible: changer de couteau
- Critère d'optimisation:

Débit	Maximiser le nombre de tâches faites
Latence	Minimiser les temps de réponse à des évènements
Équité	Faire en sorte que toutes les tâches aient accès aux ressources

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes
- Coût de *migration* (d'une ressource à une autre)
  - Coût élevé: déménager une expérience d'une ligne de lumière au synchrotron soleil
  - Coût faible: changer de couteau
- Critère d'optimisation:

Débit	Maximiser le nombre de tâches faites
Latence	Minimiser les temps de réponse à des évènements
Équité	Faire en sorte que toutes les tâches aient accès aux ressources
	Autres: Consommation énergétique, etc.

Critères de politiques d'ordonnancement

- Coût de *préemption* (pour interrompre un travail)
  - Coût élevé: Sauvegarder l'état d'un cluster de calcul haute-performance en plein milieu
  - Coût faible: Arrêter de couper les carottes le temps d'égoutter les pâtes
- Coût de *migration* (d'une ressource à une autre)
  - Coût élevé: déménager une expérience d'une ligne de lumière au synchrotron soleil
  - Coût faible: changer de couteau
- Critère d'optimisation:

Débit	Maximiser le nombre de tâches faites
Latence	Minimiser les temps de réponse à des évènements
Équité	Faire en sorte que toutes les tâches aient accès aux ressources
	Autres: Consommation énergétique, etc.

- Présence de contraintes temps-réel?
- Contraintes d'importance/criticité des travaux?
- Connaissance des durées d'exécution des travaux?

Ordonnancement non-préemptif simple:

Exemple

Une imprimante est partagée entre 20 salariés. Dans quel ordre faire les impressions?

Ordonnancement non-préemptif simple: FCFS

Exemple

Une imprimante est partagée entre 20 salariés. Dans quel ordre faire les impressions?

First Come, First Served: traitement des tâches dans l'ordre d'arrivée.

- \+ Très simple (queue FIFO)
- \+ Optimal si il y a une seule ressource
  - Maximise le débit/minimise le temps de complétion de tous les travaux
- \+ Pas de préemption ni de migration nécessaire
- \+ Besoin d'aucune connaissance sur les travaux
- \+ Pas de *famine* si tous les travaux sont finis
- – Pas de prise en compte de priorités, de contraintes de temps, de dépendances entre travaux...
- – Pour ordonnancer des threads:
  - Pas de parallélisation du CPU et des IOs
  - Latence et temps de réponse important (ne convient pas à des tâches interactives)

Ordonnancement non-préemptif simple: FCFS

Exemple

Une imprimante est partagée entre 20 salariés. Dans quel ordre faire les impressions?

First Come, First Served: traitement des tâches dans l'ordre d'arrivée.

- \+ Très simple (queue FIFO)
- \+ Optimal si il y a une seule ressource
  - Maximise le débit/minimise le temps de complétion de tous les travaux
- \+ Pas de préemption ni de migration nécessaire
- \+ Besoin d'aucune connaissance sur les travaux
- \+ Pas de *famine* si tous les travaux sont finis
- – Pas de prise en compte de priorités, de contraintes de temps, de dépendances entre travaux...
- – Pour ordonnancer des threads:
  - Pas de parallélisation du CPU et des IOs
  - Latence et temps de réponse important (ne convient pas à des tâches interactives)

Plan d'ordonnancement statique:

Exemple

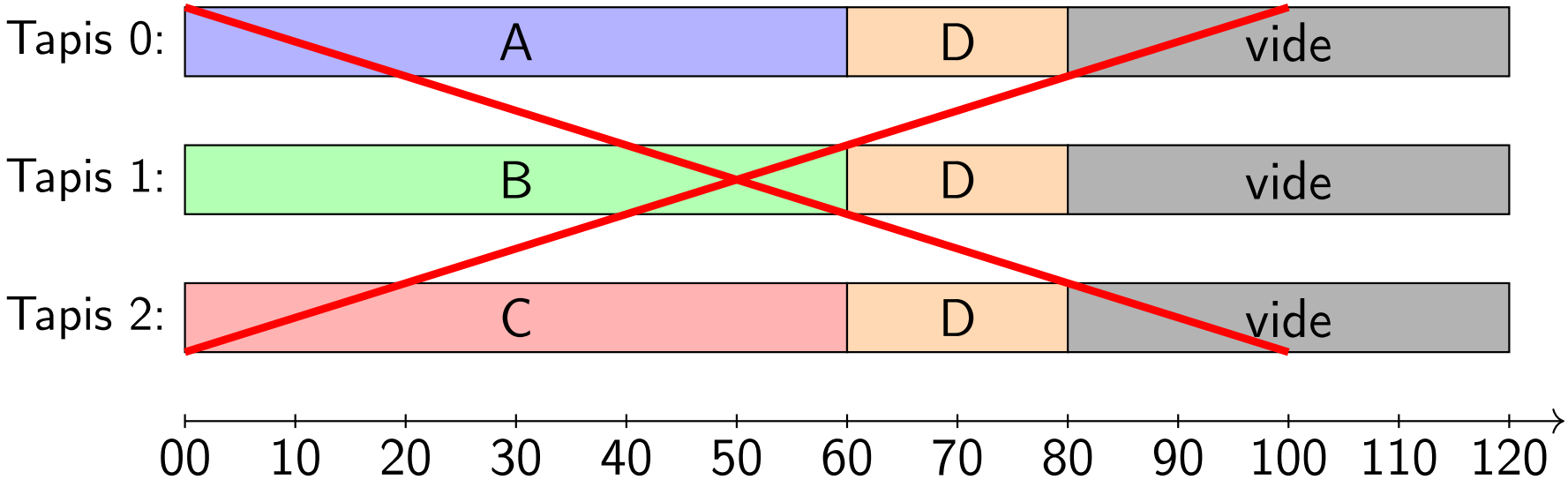
4 étudiants veulent faire 60 abdominaux chacun (rhythme: 1 par seconde) en y passant un minimum de temps. Il n'y a que 3 tapis. Comment faire?

Plan d'ordonnancement statique: Mc Naughton

Exemple

4 étudiants veulent faire 60 abdominaux chacun (rhythme: 1 par seconde) en y passant un minimum de temps. Il n'y a que 3 tapis. Comment faire?

L'étudiant D ne peut pas occuper simultanément les trois tapis!

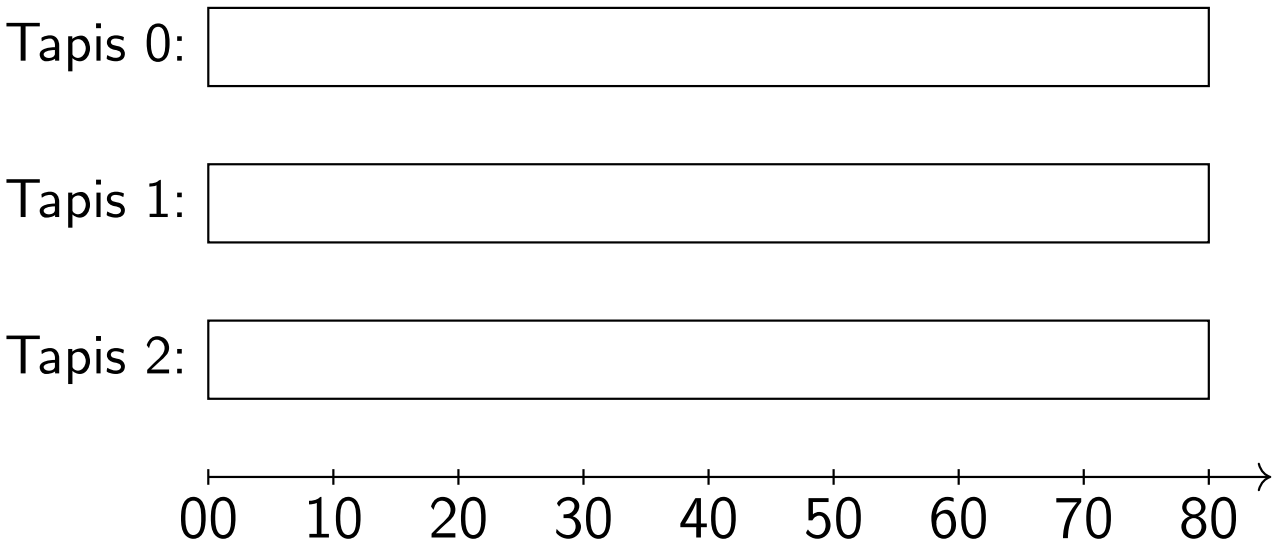


Plan d'ordonnancement statique: Mc Naughton

Exemple

4 étudiants veulent faire 60 abdominaux chacun (rhythme: 1 par seconde) en y passant un minimum de temps. Il n'y a que 3 tapis. Comment faire?

- Il y a 4*60 = 240s de travail à faire.
- En utilisant pleinement les 3 tapis, le temps minimum pour tout faire est de 240/3 = 80s.
- Algorithme: préparer le plan "en faisant des retours à la ligne"

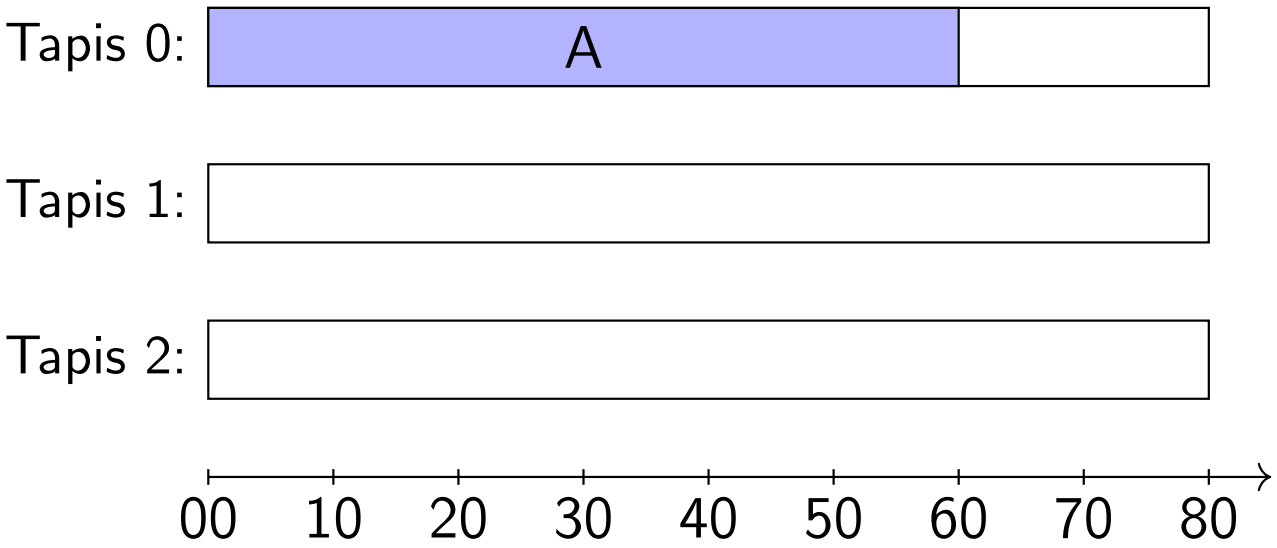


Plan d'ordonnancement statique: Mc Naughton

Exemple

4 étudiants veulent faire 60 abdominaux chacun (rhythme: 1 par seconde) en y passant un minimum de temps. Il n'y a que 3 tapis. Comment faire?

- Il y a 4*60 = 240s de travail à faire.
- En utilisant pleinement les 3 tapis, le temps minimum pour tout faire est de 240/3 = 80s.
- Algorithme: préparer le plan "en faisant des retours à la ligne"

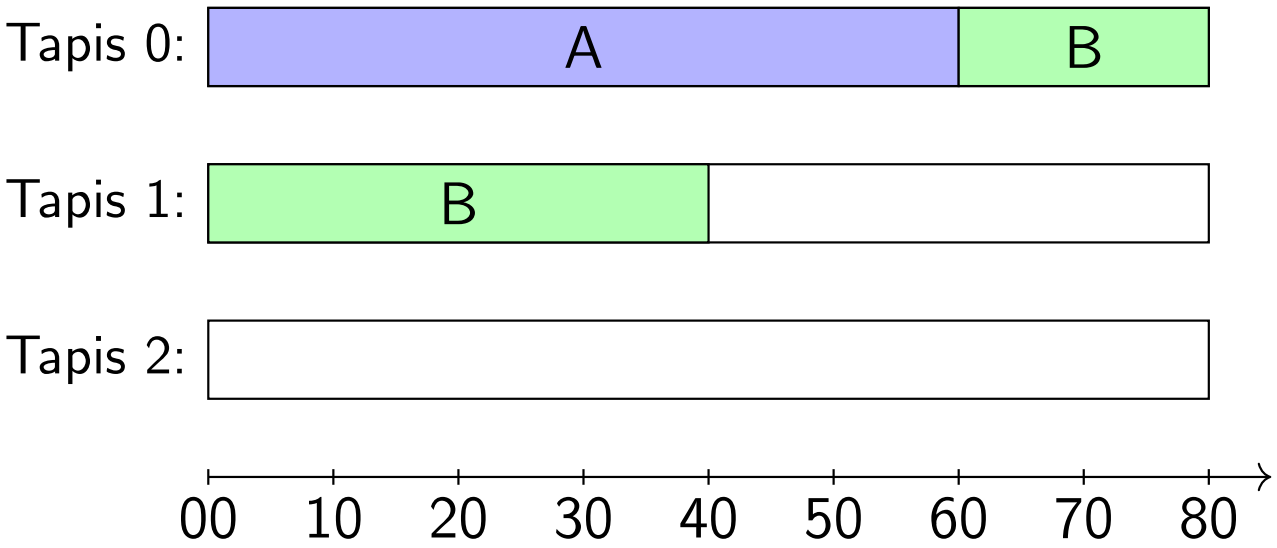


Plan d'ordonnancement statique: Mc Naughton

Exemple

4 étudiants veulent faire 60 abdominaux chacun (rhythme: 1 par seconde) en y passant un minimum de temps. Il n'y a que 3 tapis. Comment faire?

- Il y a 4*60 = 240s de travail à faire.
- En utilisant pleinement les 3 tapis, le temps minimum pour tout faire est de 240/3 = 80s.
- Algorithme: préparer le plan "en faisant des retours à la ligne"

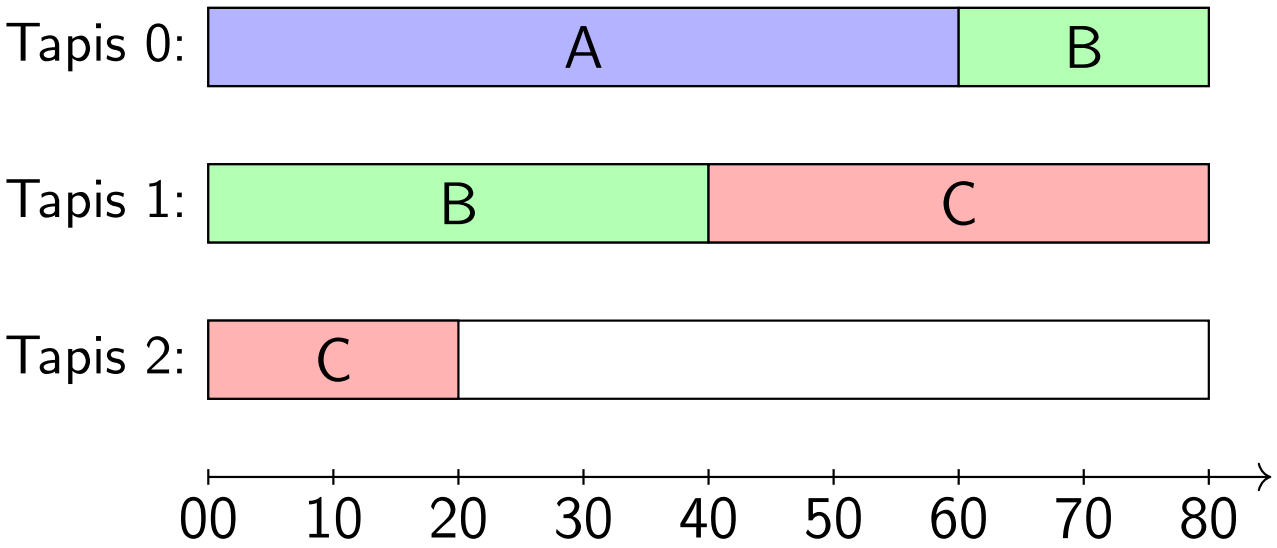


Plan d'ordonnancement statique: Mc Naughton

Exemple

4 étudiants veulent faire 60 abdominaux chacun (rhythme: 1 par seconde) en y passant un minimum de temps. Il n'y a que 3 tapis. Comment faire?

- Il y a 4*60 = 240s de travail à faire.
- En utilisant pleinement les 3 tapis, le temps minimum pour tout faire est de 240/3 = 80s.
- Algorithme: préparer le plan "en faisant des retours à la ligne"

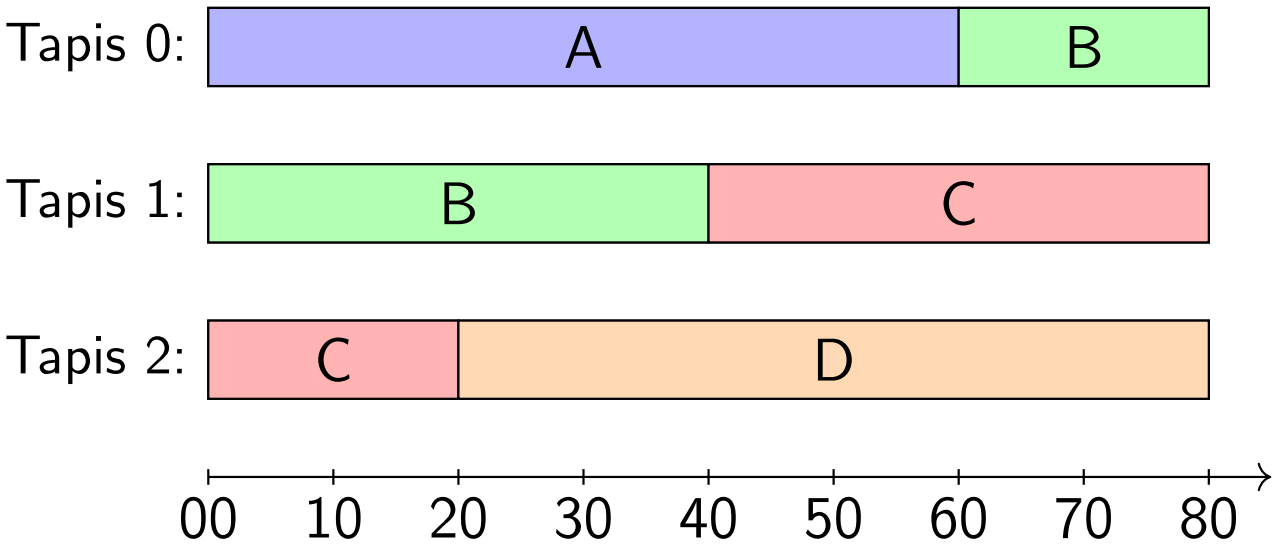


Plan d'ordonnancement statique: Mc Naughton

Exemple

4 étudiants veulent faire 60 abdominaux chacun (rhythme: 1 par seconde) en y passant un minimum de temps. Il n'y a que 3 tapis. Comment faire?

- Il y a 4*60 = 240s de travail à faire.
- En utilisant pleinement les 3 tapis, le temps minimum pour tout faire est de 240/3 = 80s.
- Algorithme: préparer le plan "en faisant des retours à la ligne"



Plan d'ordonnancement statique: Mc Naughton

Exemple

4 étudiants veulent faire 60 abdominaux chacun (rhythme: 1 par seconde) en y passant un minimum de temps. Il n'y a que 3 tapis. Comment faire?

- − Nécessite de faire des *migrations*
- − Nécessite de connaitre tous les travaux et leurs durée en avance
- \+ Optimal (solution qui prend le temps minimal) sur multi-processeur

Algorithme

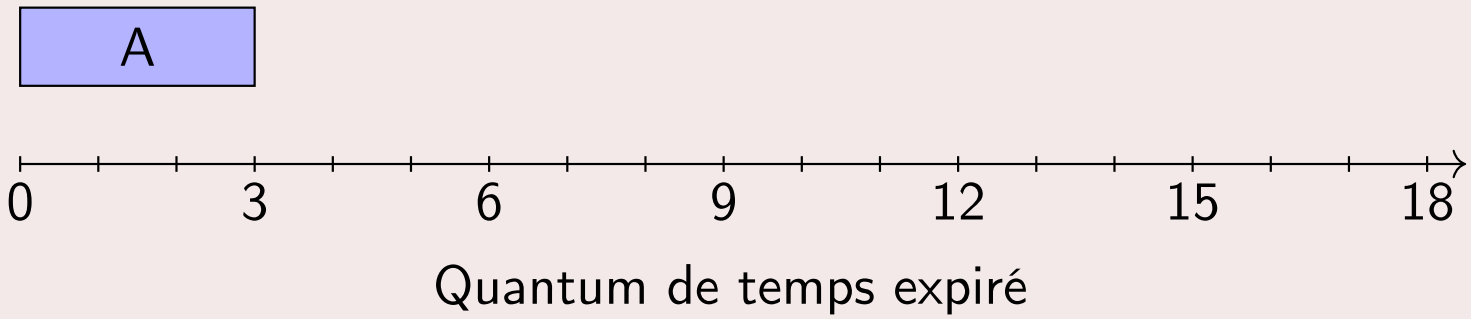
Exemple

Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?

Algorithme du tourniquet (round-robin)

Exemple

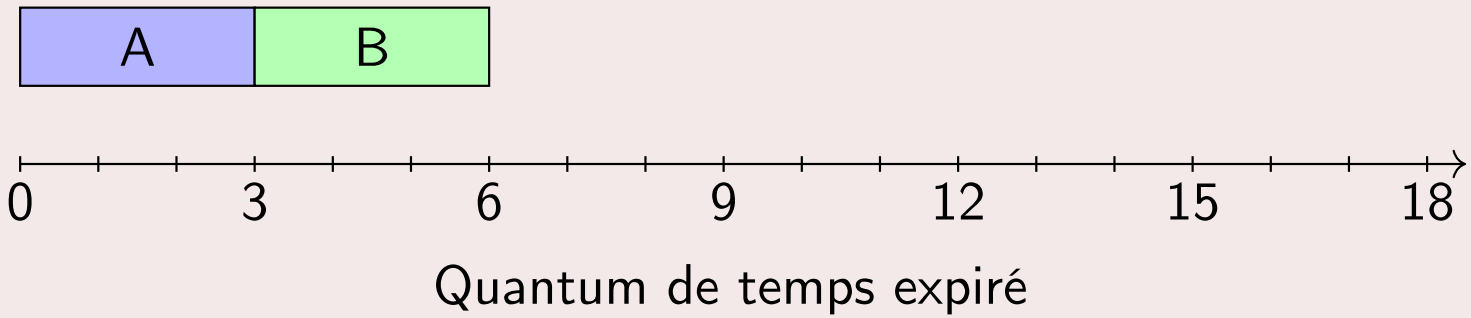
Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?



Algorithme du tourniquet (round-robin)

Exemple

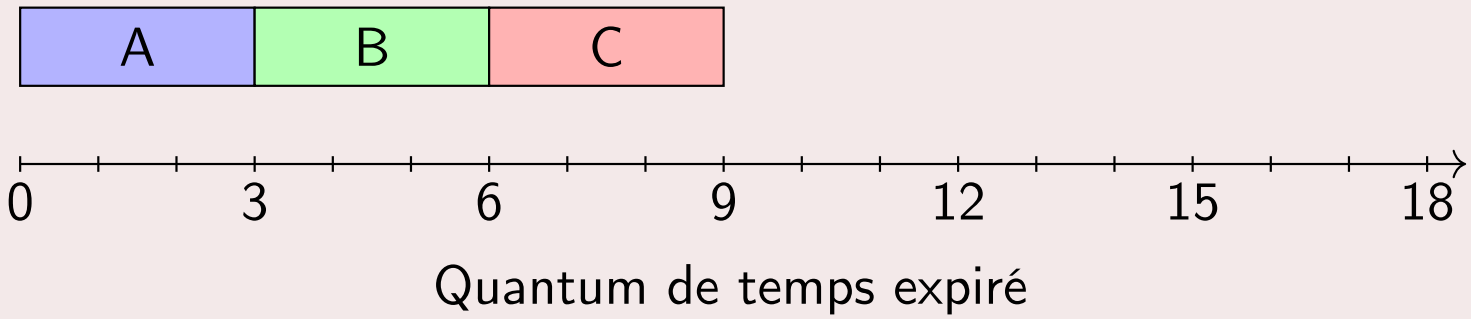
Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?



Algorithme du tourniquet (round-robin)

Exemple

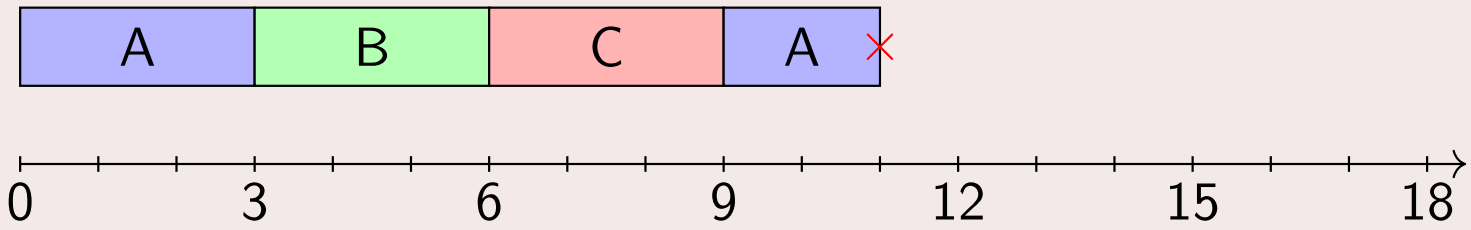
Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?



Algorithme du tourniquet (round-robin)

Exemple

Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?

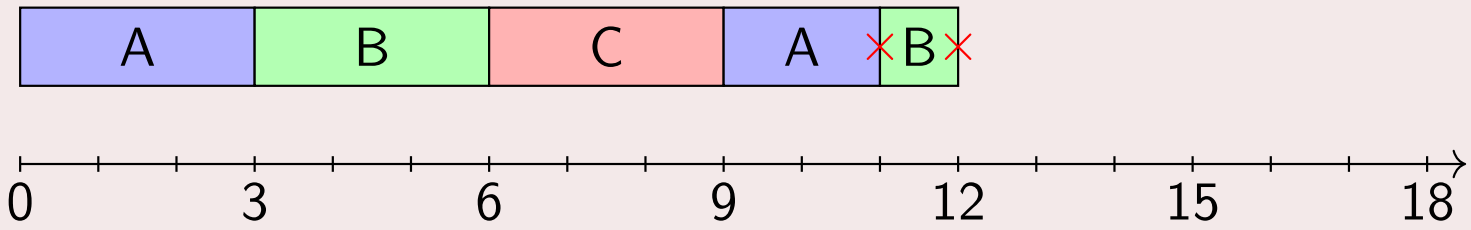


Le thread bloque

Algorithme du tourniquet (round-robin)

Exemple

Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?

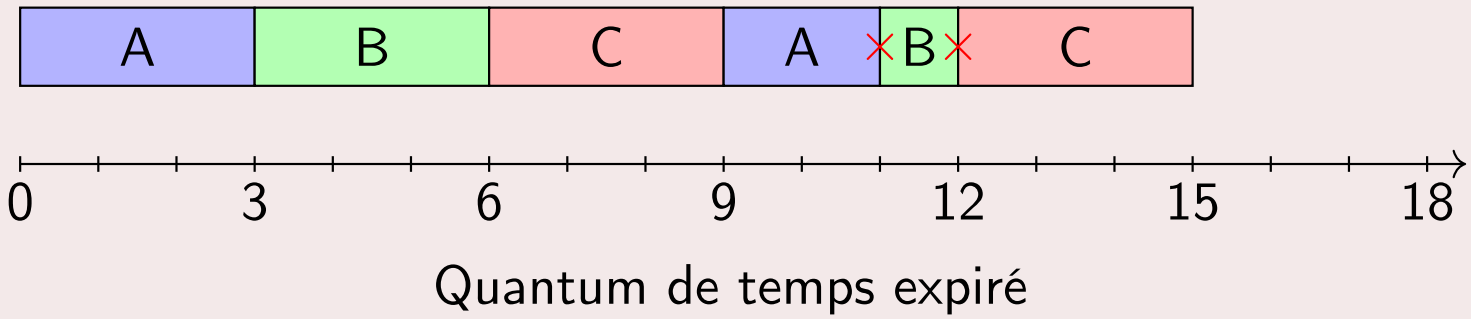


Le thread bloque

Algorithme du tourniquet (round-robin)

Exemple

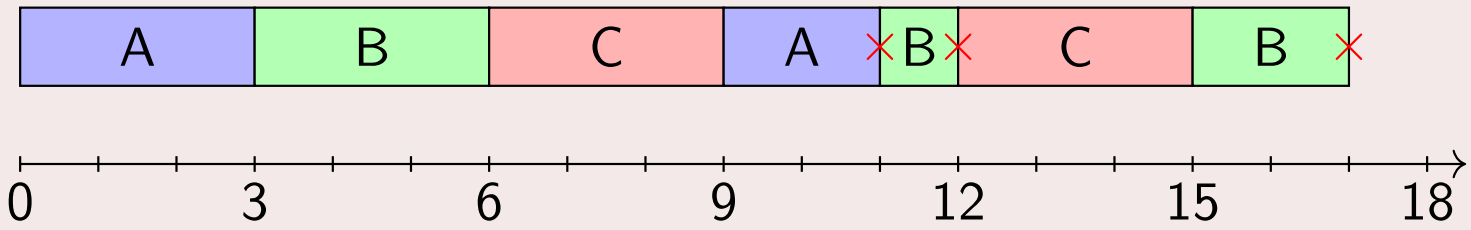
Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?



Algorithme du tourniquet (round-robin)

Exemple

Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?

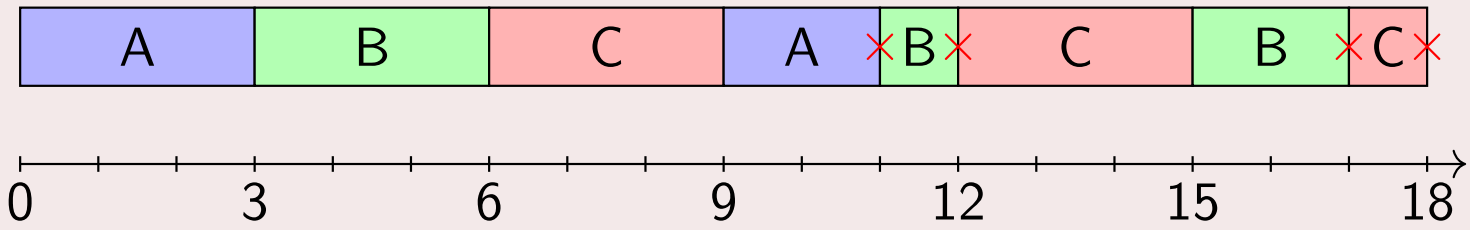


A est toujours bloqué puis B bloque

Algorithme du tourniquet (round-robin)

Exemple

Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?

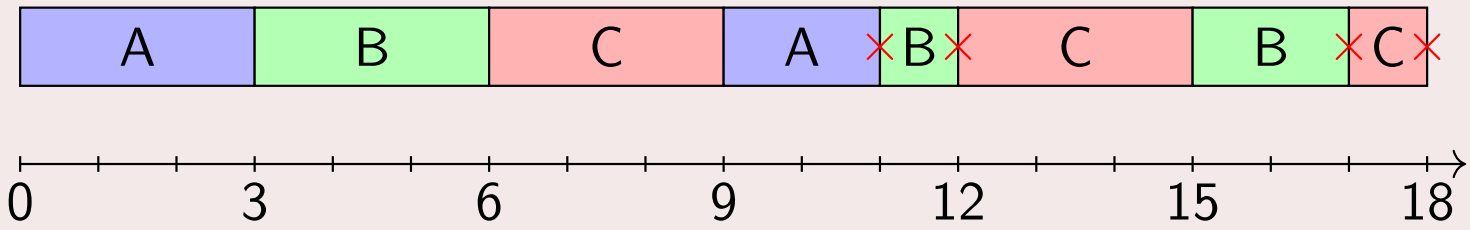


Le thread bloque

Algorithme du tourniquet (round-robin)

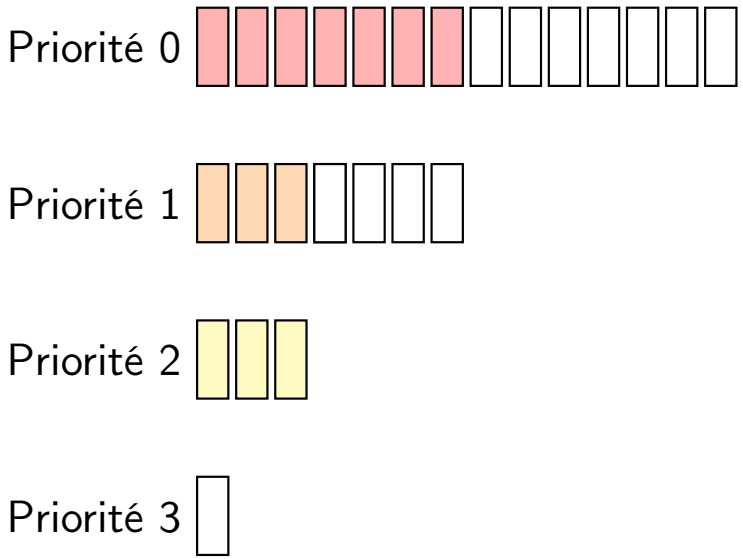
Exemple

Vous êtes seul et devez nourrir trois bébés. Lorsqu'un bébé attend sa cuillèrée trop longtemps, il pleure et vous n'aimez pas ça. Dans quel ordre nourrissez vous les bébés?



- \+ Faible temps d'attente par processus
- \+ Équité de traitement entre les processus
- \+ Le découpage en quantums permet de paralléliser les IOs et le CPU
- \+ Le découpage en quantums permet d'équilibrer la charger en multiprocesseurs
- – Les multiples préemptions prennent du temps CPU
  - ➜ Difficulté du choix du quantum de temps: compromis coût/réactivité.

Multi-level feedback queue



- Amélioration de l'algorithme du tourniquet avec plusieurs files:
  - Quand un thread est bloqué ou attends depuis trop longtemps, il remonte dans les priorités
  - Quand un thread utilise tout son quota de temps, il descend

➜ Permet de prioriser les threads "I/O-bound" sur ceux "CPU bound".

Ordonnancement a priorité fixe

Exemple

Il y a 3 chefs au dessus de vous dans l'organigramme de votre société ($N+1, N+2, N+3$). Chacun vous demande de temps en temps des travaux à faire. Le but de votre vie est d'arriver au sommet au plus vite. Dans quel ordre faites-vous ces travaux?

Ordonnancement a priorité fixe

Exemple

Il y a 3 chefs au dessus de vous dans l'organigramme de votre société ($N+1, N+2, N+3$). Chacun vous demande de temps en temps des travaux à faire. Le but de votre vie est d'arriver au sommet au plus vite. Dans quel ordre faites-vous ces travaux?

- Ordonnancement à priorité fixe
  - À chaque travail est affecté une priorité
  - À tout moment est exécuté une des tâches dont la priorité la plus forte
  - Si une nouvelle tâche de priorité plus forte que l'actuelle arrive: il y a préemption.
- Exemple déjà recontré:

Ordonnancement a priorité fixe

Exemple

Il y a 3 chefs au dessus de vous dans l'organigramme de votre société ($N+1, N+2, N+3$). Chacun vous demande de temps en temps des travaux à faire. Le but de votre vie est d'arriver au sommet au plus vite. Dans quel ordre faites-vous ces travaux?

- Ordonnancement à priorité fixe
  - À chaque travail est affecté une priorité
  - À tout moment est exécuté une des tâches dont la priorité la plus forte
  - Si une nouvelle tâche de priorité plus forte que l'actuelle arrive: il y a préemption.
- Exemple déjà recontré: les interruptions
  - Les interrupt handlers sont prioritaires sur les thread normaux

Ordonnancement temps-réel

Définition (Système temps-réel)

Un système temps-réel est un système qui a la capacité de répondre à des évènements asynchrones issus du monde extérieur dans des délais pré-déterminés.
Exemples: contrôle-commande, systèmes financiers, télécommunication.

- Temps-réel $\neq$ rapide!
- Pour réaliser un système, on va généralement le découper en différents threads, avec des contraintes de temps différentes.
- Souvent: on divise le système en deux types de tâches:

  Périodiques Déclenchement sur timer, la durée entre 2 travaux est fixe, échéance = durée

  Sporadiques Déclenchement sur interruption, il y a une durée minimale entre 2 travaux, échéance = cette durée

Ordonnancement en priorité fixe pour le temps-réel

Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 0.66 s
- B, de période 3 s et de durée de calcul max. 1.5 s

On souhaite utiliser un ordonnancement en priorité fixe. Comment affecter les priorités?

Ordonnancement en priorité fixe pour le temps-réel

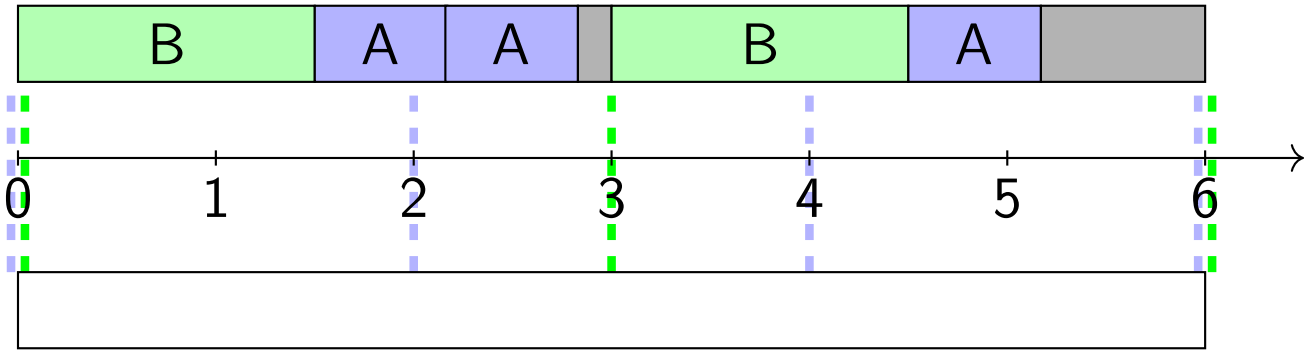
Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 0.66 s
- B, de période 3 s et de durée de calcul max. 1.5 s

On souhaite utiliser un ordonnancement en priorité fixe. Comment affecter les priorités?

B prioritaire ✗



Ordonnancement en priorité fixe pour le temps-réel

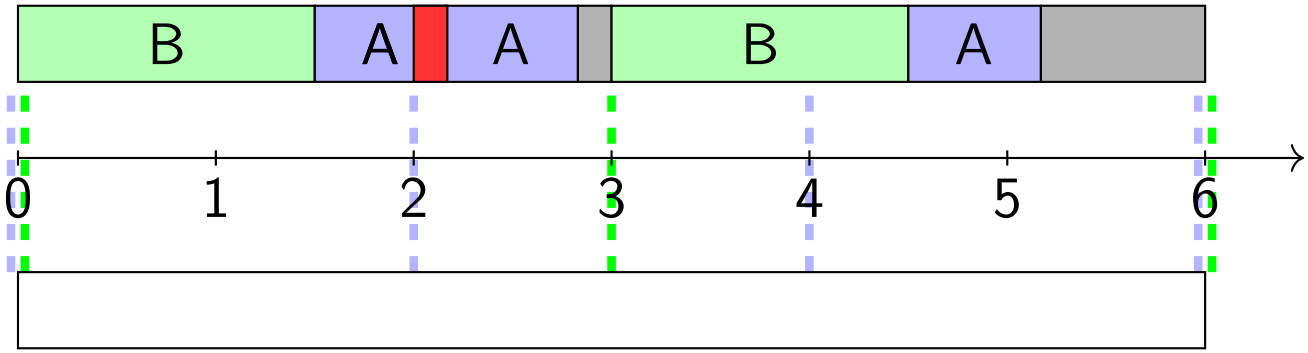
Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 0.66 s
- B, de période 3 s et de durée de calcul max. 1.5 s

On souhaite utiliser un ordonnancement en priorité fixe. Comment affecter les priorités?

B prioritaire ✗



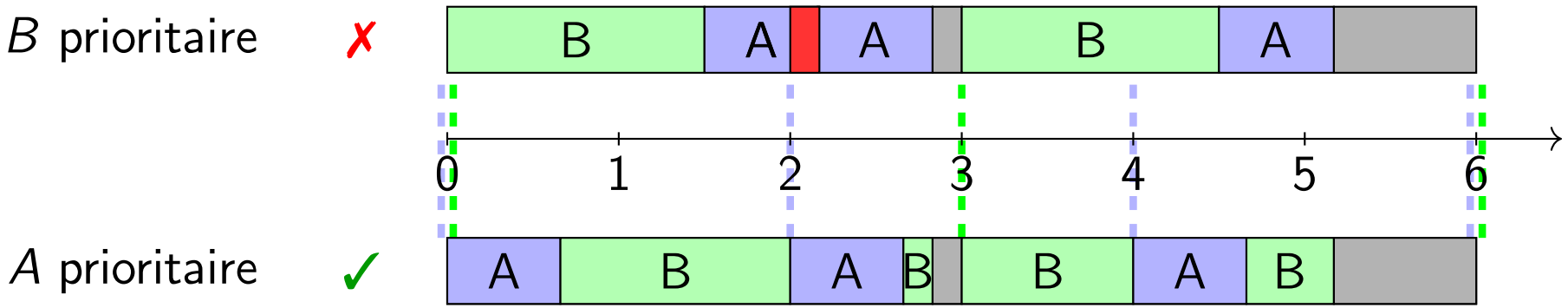
Ordonnancement en priorité fixe pour le temps-réel

Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 0.66 s
- B, de période 3 s et de durée de calcul max. 1.5 s

On souhaite utiliser un ordonnancement en priorité fixe. Comment affecter les priorités?



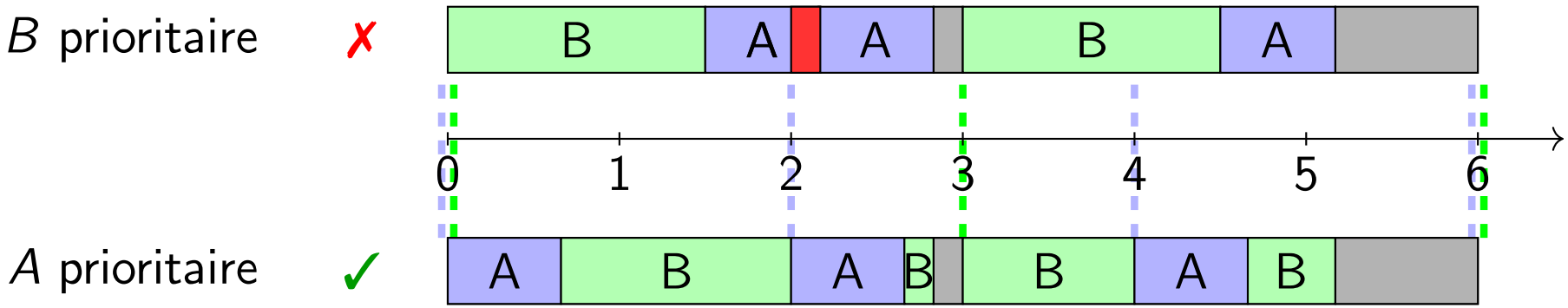
Ordonnancement en priorité fixe pour le temps-réel

Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 0.66 s
- B, de période 3 s et de durée de calcul max. 1.5 s

On souhaite utiliser un ordonnancement en priorité fixe. Comment affecter les priorités?



Théorème (Liu & Layland)

Si on peut ordonnancer les tâches en priorité fixe, on saure le faire en affectant les priorités dans l'ordre inverse des périodes.

Ordonnancement temps-réel optimal

Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 1 s
- B, de période 3 s et de durée de calcul max. 1.5 s

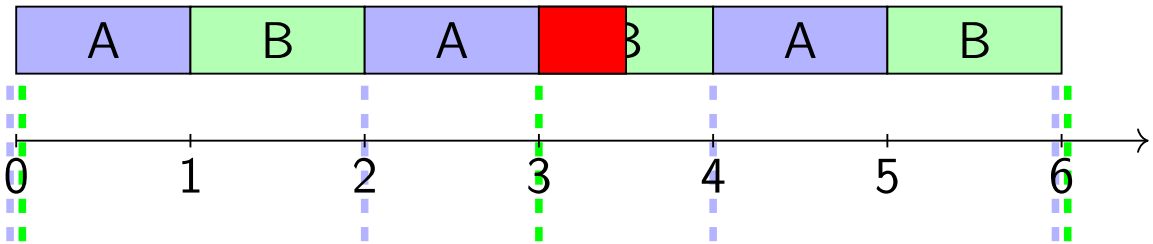
Ordonnancement temps-réel optimal

Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 1 s
- B, de période 3 s et de durée de calcul max. 1.5 s

Priorité fixe ✗



Théorème (Liu & Layland)

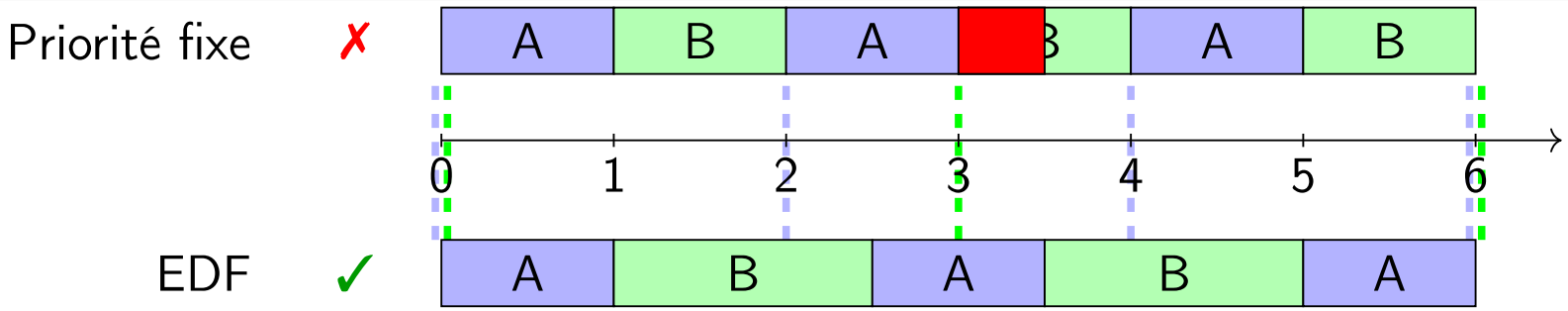
Si la charge du système est $< 69\%$, alors les tâches sont ordonnançables en priorité fixe.

Ordonnancement temps-réel optimal

Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 1 s
- B, de période 3 s et de durée de calcul max. 1.5 s



Théorème (Liu & Layland)

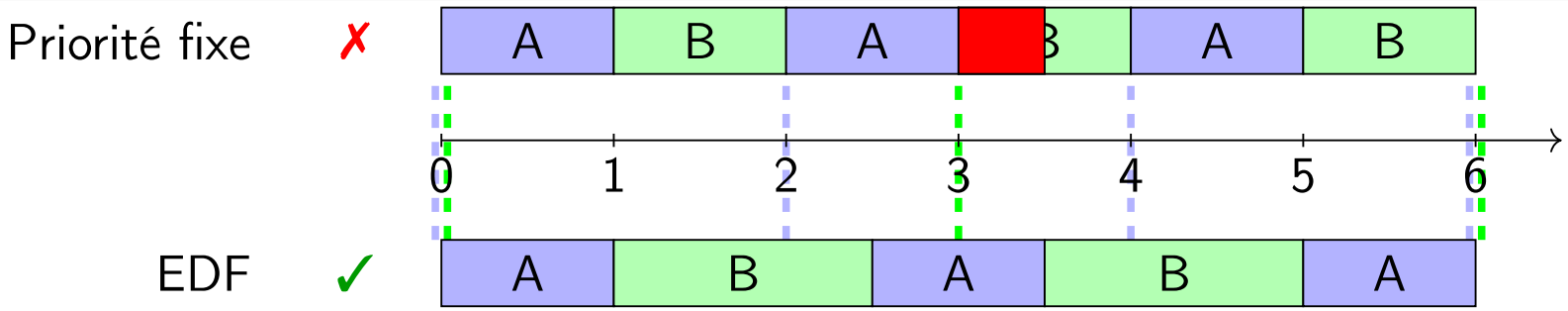
En monoprocesseur, tout système de tâche périodique faisable (charge processeur $< 100\%$) est faisable avec l'algorithme EDF.

Ordonnancement temps-réel optimal

Exemple

Un système doit exécuter deux tâches périodiques:

- A, de période 2 s et de durée de calcul max. 1 s
- B, de période 3 s et de durée de calcul max. 1.5 s



Théorème (Liu & Layland)

En monoprocesseur, tout système de tâche périodique faisable (charge processeur $< 100\%$) est faisable avec l'algorithme EDF.

- EDF reste optimal sur des tâches non périodiques
- Connaissance des durées de calcul des tâches non nécessaire
- Il n'existe pas d'ordonnancement optimal en multi-processeur (sauf si tous les paramètres des tâches sont connues à l'avance)

Sommaire

1. Cours 4: Interruptions; politiques d'ordonnancement
   - Interruptions
   - Ordonnancement
   - Conclusion

Conclusion

- Mécanisme d'interruption: prévenir le processeur d'un évènement urgent à traiter
- Permet de mettre en place des préemptions
- Différentes stratégies d'ordonnancement
  - Préemptif ou non-préemptif
  - Division en quantum de temps; interactivité versus surcoût
  - Prise en compte de contraintes temps-réel

Classification des d'algorithmes d'ordonnancement

Statique le plan d'ordonnancement est décidé avant l'exécution (ex: McNaughton)

Dynamique le plan d'ordonnancement est décidé pendant l'exécution

- **Non préemptif** on ne peut pas interrompre les jobs (ex: FCFS)
- **Préemptif** on peut
  - **Priorité statique** la priorité des travaux ne change pas (RM, round-robin)
  - **Priorité dynamique** la priorité des travaux peut changer (EDF,multi-level feedback)