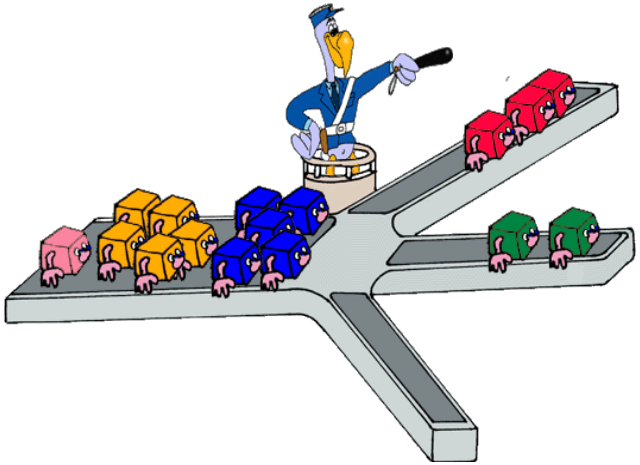
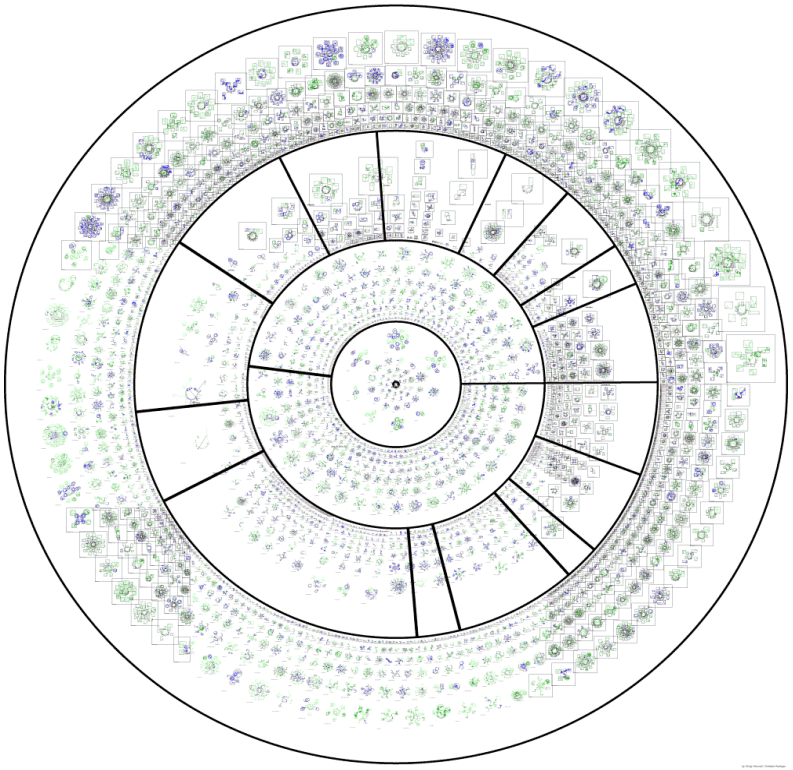
Programmation Système : ordonnancement



Linux Kernel v2.4.9

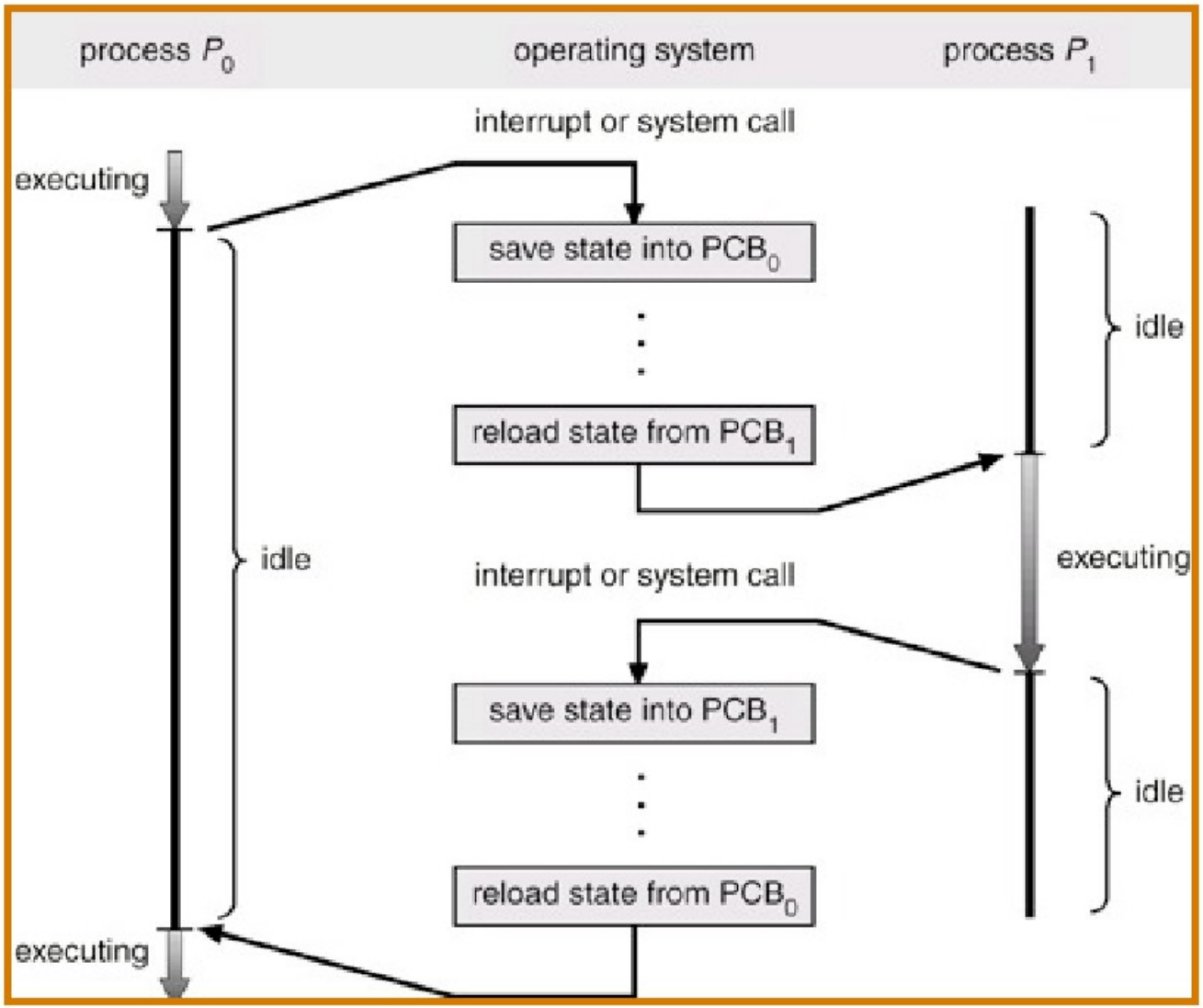


Rappel : la préemption

- La **préemption** se définit comme la réquisition du processeur pour l'exécution d'une tâche et d'une seule pendant un temps déterminé.

- Dans un ordonnancement (statique à base de priorités) avec préemption : lorsque le processeur est inactif, la tâche prête de plus haute priorité sera choisie pour être exécutée. A chaque instant cette tâche peut être pré-emptée (remplacée) par n'importe quelle tâche plus prioritaire qui serait devenue prête.

Rappel : changement de contexte



L'ordonnanceur

- L'**ordonnanceur** (***scheduler***) désigne le composant du noyau du système d'exploitation dont le rôle est de choisir les processus qui vont être exécutés par les processeurs d'un ordinateur.

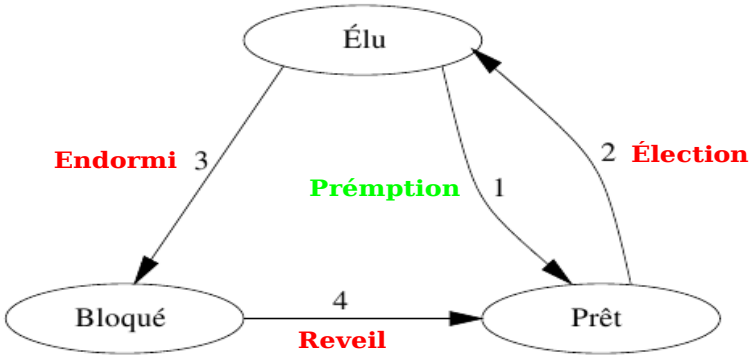
- Pour la très grande majorité des ordinateurs, avoir un seul processeur implique de ne pouvoir effectuer qu'un traitement à la fois. Or, à un instant donné, il est possible qu'il y ait plus de processus à exécuter qu'il n'y a de processeurs.

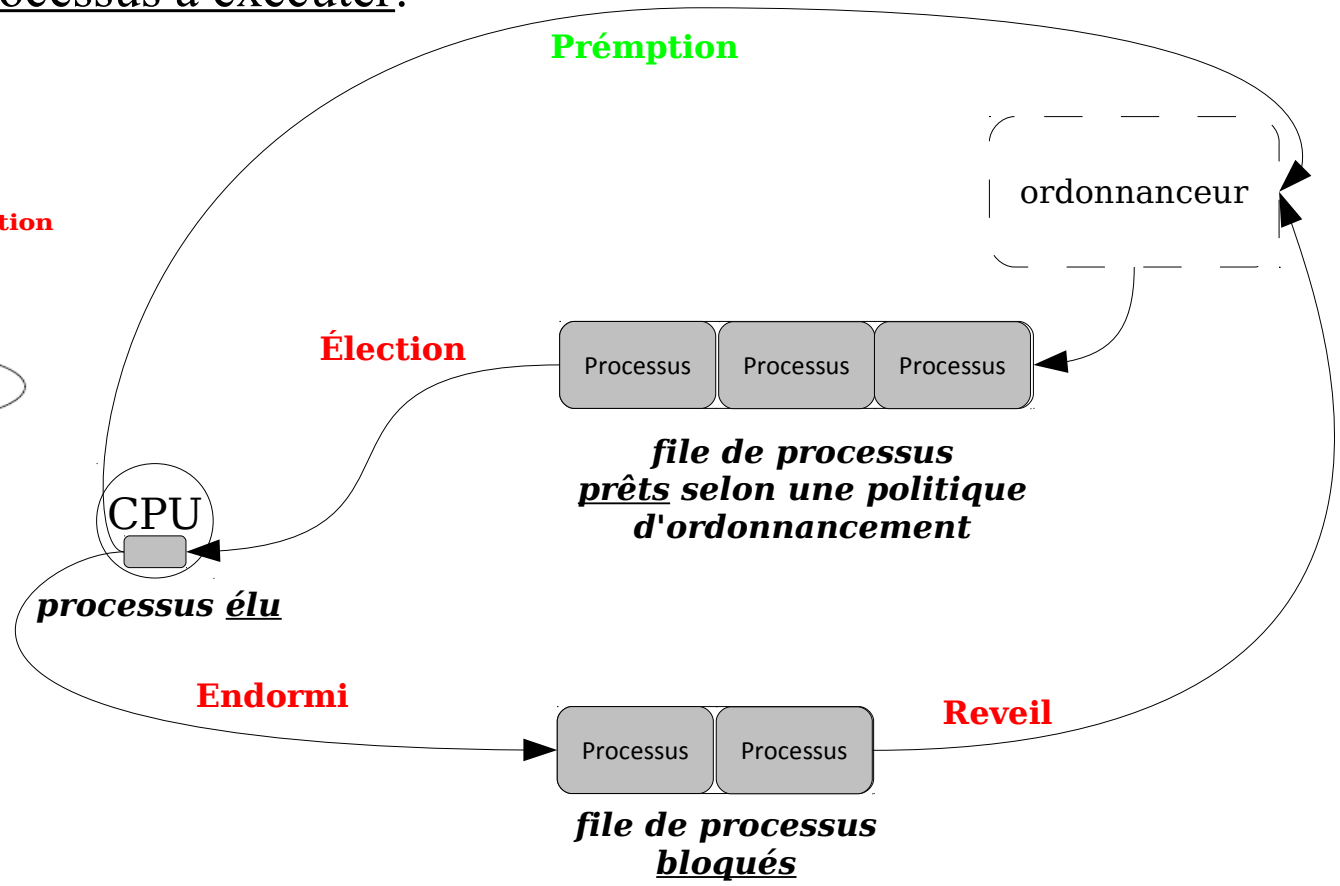
- Pour arriver à donner l'illusion que plusieurs tâches sont traitées simultanément, l'ordonnanceur du noyau du système s'appuie sur les notions de temps partagé, de commutation de contexte et d'ordonnancement.

- Un ordonnanceur fait face à deux problèmes principaux :
  - le choix du processus à exécuter, et
  - le temps d'allocation du processeur au processus choisi.

Principe (mono-processeur)

- Le rôle de l'ordonnanceur du noyau, est de permettre à tous ces processus de s'exécuter et d'utiliser le processeur de manière optimale du point de vue de l'utilisateur. Il procède de la manière suivante : à intervalles réguliers, le système appelle une procédure d'ordonnancement qui élit le processus à exécuter.

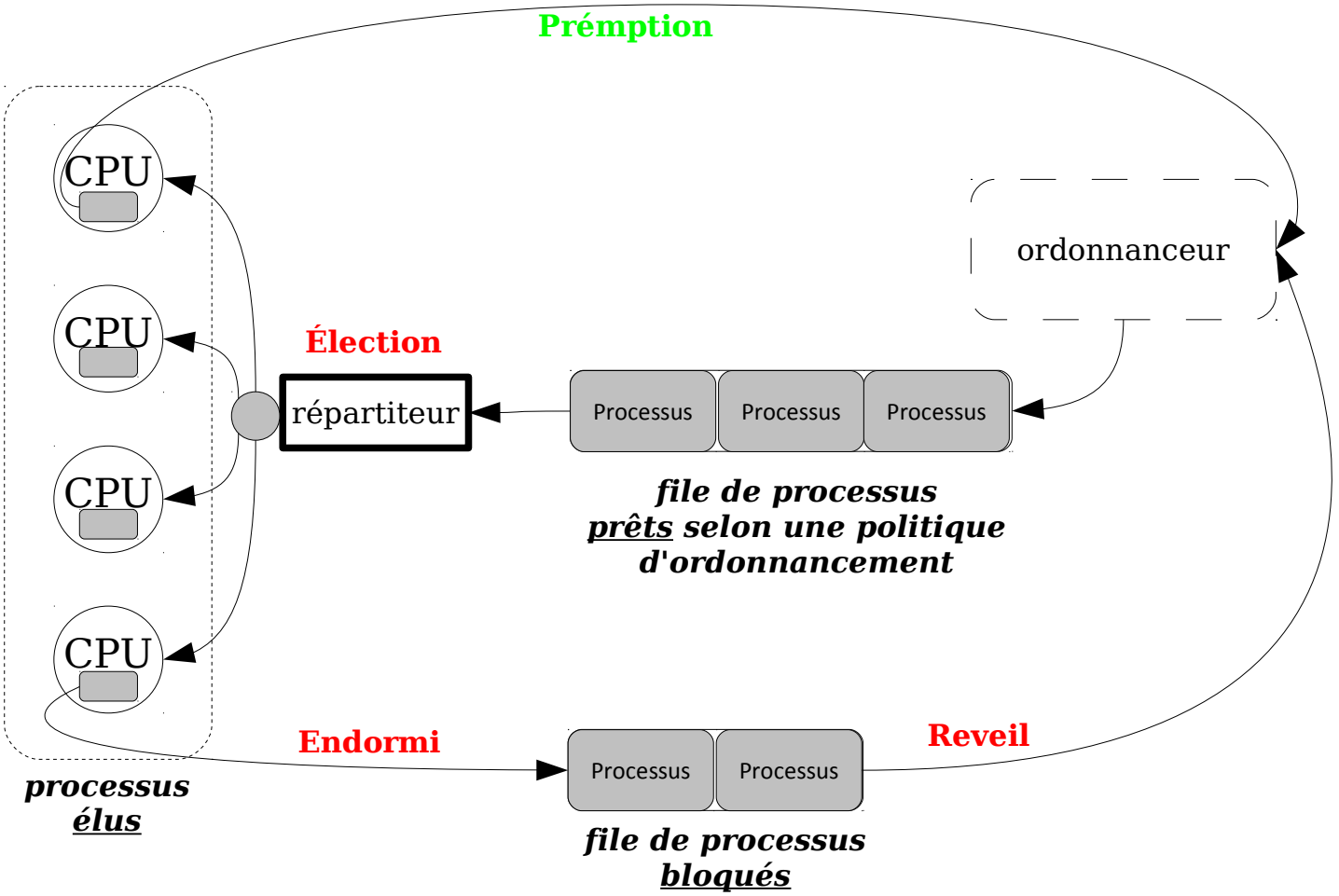




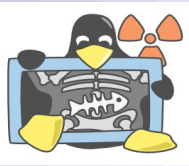
Principe (multi-processeur)

- L'**ordonnanceur (*scheduler*)** désigne le composant du noyau du système d'exploitation dont le rôle est de choisir les processus qui vont être exécutés par les processeurs d'un ordinateur.

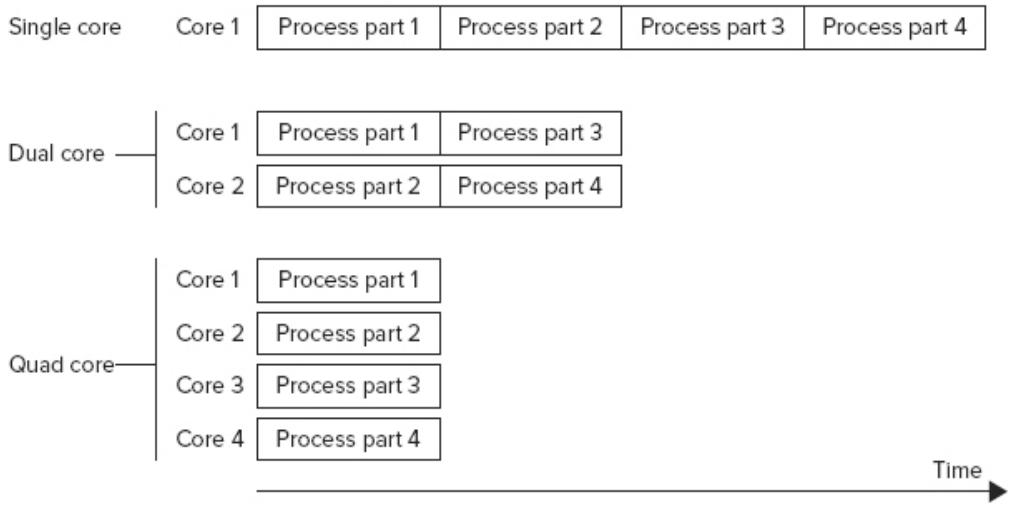
- Le noyau Linux 2.6 associe un processus avec un des coeurs (*affinity*) et gère des *file* par coeur.



Multi-coeur



- Aujourd'hui, les architectures (CPU et GPU) utilisent l'intégration de plusieurs coeurs sur un processeur unique.
- Un exemple pourrait être le traitement des pixels d'une image par un algorithme qui ne nécessite pas d'informations sur les pixels voisins. L'algorithme pourrait diviser l'image en quatre parties :
  - sur un processeur *single-core*, chaque partie est traitée de façon séquentielle
  - sur un CPU *dual-core*, deux pièces sont traitées en parallèle,
  - et sur un processeur *quad-core*, quatre pièces sont traitées en parallèle entraînant une échelle quasi-linéaire de la performance avec le nombre de cœurs.



Classes et types d'ordonnancement

- Il existe deux grandes classes d'ordonnancement :
  - L'**ordonnancement en temps partagé** : le processeur passe donc d'un processus à un autre en exécutant chaque processus pendant quelques dizaines ou centaines de millisecondes. Le temps d'allocation du processeur au processus est appelé ***quantum***. C'est l'ordonnancement utilisé sur la plupart des systèmes classiques.
  - L'**ordonnancement en temps réel** : permettra de respecter des contraintes de temps indépendamment de sa charge (système dit « déterministe »). Ce type d'ordonnancement est généralement utilisé dans les systèmes embarqués.
- Il existe deux grands types d'algorithmes d'ordonnancement :
  - **Ordonnancement circulaire (*round robin*)**
  - **Ordonnancement avec priorité**

Ordonnancement

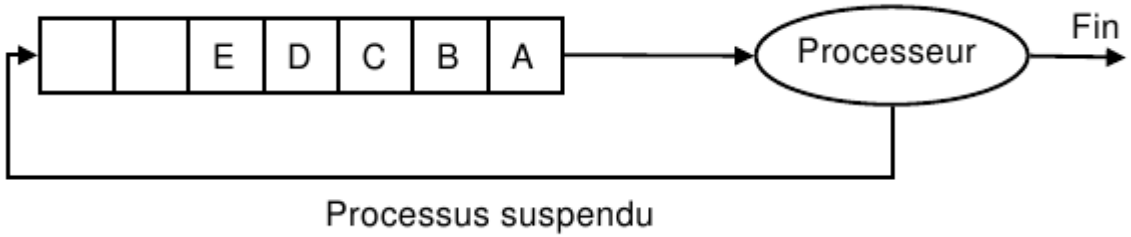
- Les objectifs de l'ordonnancement d'un système sont, entre autres :
  - S'assurer que chaque processus reçoive sa part de temps processeur
  - Minimiser le temps de réponse
  - Utiliser le processeur à 100%
  - Utilisation équilibrée des ressources
  - Prendre en compte des priorités
  - Être prédictible

- Un **algorithme d'ordonnancement** sert à choisir lequel de plusieurs processus sera traité en premier par le processeur.

- *Rappel : Un système d'exploitation multitâche est préemptif lorsque celui-ci peut arrêter (réquisition) à tout moment n'importe quel processus pour passer la main suivant.*

Ordonnancement circulaire

- L'algorithme du tourniquet circulaire (ou *round robin*) est un algorithme ancien, simple, fiable et très utilisé.
- Il mémorise dans une file du type FIFO (*First In First Out*) la liste des processus prêts, c'est-à-dire en attente d'exécution.
- Il alloue le processeur au processus en tête de file, pendant un quantum de temps.
- L'algorithme round robin permet une répartition équitable du processeur.
- Le choix de la valeur du quantum se révèle très importante :
  - Un quantum trop petit provoque trop de commutations de processus et abaisse l'efficacité du processeur
  - Un quantum trop élevé augmente le temps de réponse des courtes commandes en mode interactif
- Un quantum entre 10 et 50 ms est souvent un compromis raisonnable.



Ordonnancement avec priorité

- L'ordonnanceur à priorité attribue à chaque processus une priorité.
- Le choix du processus à élire dépend donc des priorités des processus prêts.
- Les processus de même priorité sont regroupés dans une file du type FIFO. Il y a autant de files qu'il y a de niveaux de priorité. L'ordonnanceur choisit le processus le plus prioritaire qui se trouve en tête de file.
- En général, les processus de même priorité sont ordonnancés selon l'algorithme du tourniquet.
- Pour empêcher les processus de priorité élevée de s'exécuter indéfiniment, l'ordonnanceur diminue régulièrement la priorité du processus en cours d'exécution.

Ordonnancement temps réel

- Théoriquement, le concepteur d'un système temps réel devrait être capable de prouver que les limites temporelles ne seront jamais dépassées quelle que soit la situation. Cette vérification est appelée "test d'acceptabilité" et dépend de l'ordonnanceur utilisé et des caractéristiques des tâches du système.

- Il existe de nombreux algorithmes d'ordonnancement :
  - *Rate-monotonic scheduling* (RMS) : attribue la priorité la plus forte à la tâche qui possède la plus petite période (RMS est optimal dans le cadre d'un système temps réel préemptif de tâches périodiques)

  - *Earliest deadline first scheduling* (EDF) : attribue une priorité à chaque requête en fonction de l'échéance de cette dernière (utilisé dans les systèmes préemptifs à priorité dynamique utilisé dans les systèmes temps réel)

  - *Shortest job first* (« plus court processus en premier ») : attribue le processeur au plus court des processus de la file d'attente (mais nécessite une évaluation précise du temps d'exécution de tous les processus en attente de traitement).

Ordonnanceur Windows

- L'algorithme d'ordonnancement de Windows est basé sur une priorité qui peut changer au cours de l'exécution des processus.

- À chaque valeur de priorité une file d'attente est associée et Windows parcourt ces files de manière classique.

- Chaque file est traitée par l'algorithme du tourniquet.

Ordonnanceur Linux

- Linux utilise un algorithme de temps partagé basé sur des valeurs de priorité.
- Premièrement, chaque processus possède une valeur de priorité (statique) qui lui est attribuée au moment de sa création.
- Puis, l'ordonnanceur attribue à chaque processus un quantum initial (normalement avec un pas de 10ms) qui est égale à la valeur de priorité. Ainsi, un processus de priorité 20 aura un quantum de 200 ms. À chaque 10 ms, on diminue de 1 la valeur du quantum du processus en cours d'exécution dans le processeur. Chaque fois que l'ordonnanceur est appelé, une note est attribuée à tous les processus. Cette note dépend à la fois de la priorité du processus et de la valeur actuelle de son quantum (Note = Quantum + Priorité). C'est cette note (la plus forte) qui permettra de déterminer quel processus prendra le contrôle du processeur.
- On remarquera aussi qu'un processus qui a écoulé tout son quantum reste en attente tant qu'il y a des processus qui peuvent s'exécuter. Il ne se verra attribuer un nouveau quantum que lorsque tous les autres processus auront épuisé leur quantum ou seront bloqués.

Ordonnanceur complètement équitable

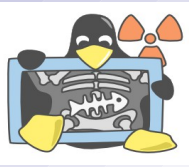
- Le *Completely Fair Scheduler* (ordonnanceur complètement équitable) est un ordonnanceur de tâches pour le noyau Linux qui a fait son apparition avec la version 2.6.23.

- Il gère l'allocation de ressource processeur pour l'exécution des processus, en maximisant l'utilisation globale du CPU tout en optimisant l'interactivité.

- Contrairement au précédent ordonnanceur utilisé par le noyau Linux, CFS n'est pas basé sur des files de processus, mais utilise un arbre binaire de recherche contenant une chronologie des futures exécutions des tâches.

- L'ordonnanceur choisit le processus le plus en manque de temps d'exécution pour tendre au mieux vers le comportement du processeur multitâche idéal.

Politique d'ordonnancement Linux



- L'appel système **`sched_setscheduler()`** permet de fixer la politique d'ordonnancement (et ses paramètres) d'un processus sous Linux.
- Actuellement, les **politiques normales** (c'est-à-dire non temps réel) proposées par Linux sont les suivantes :
  - **SCHED_OTHER** : la politique standard de temps partagé « round-robin »
  - SCHED_BATCH : pour une exécution de style traitement par lot des processus
  - SCHED_IDLE : pour l'exécution de tâches de très faible priorité en arrière-plan
- Les **politiques temps réel** suivantes sont également gérées :
  - **SCHED_FIFO** : une politique de « premier entré, premier sorti »
  - **SCHED_RR** : une politique « round-robin »

- *Remarque : le détail de chacune de ces politiques est décrite dans les pages **man**.*