Systèmes d'exploitation

Plan du cours

- Introduction aux systèmes d'exploitation
- Présentation générale d'UNIX
- Processus et parallélisme
- Ordonnancement
- Communication et synchronisation

Plan du cours

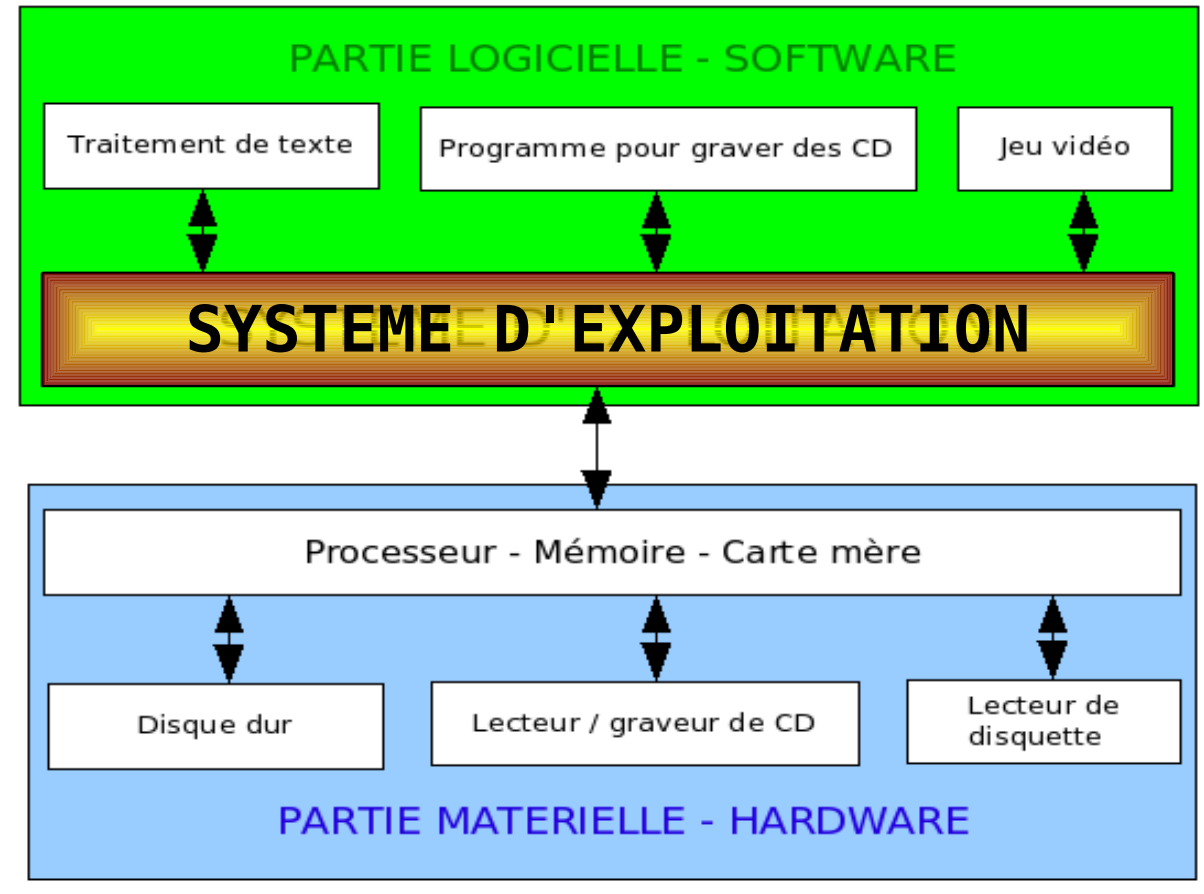
- **Introduction aux systèmes d'exploitation**
- Présentation générale d'UNIX
- Processus et parallélisme
- Ordonnancement
- Communication et synchronisation

Rôle d'un système d'exploitation (1)

Applications

Matériel

Rôle d'un système d'exploitation (2)



Rôle d'un système d'exploitation (3)

- **Logiciel** destiné à faciliter et simplifier l'utilisation d'un ordinateur
- **Interface** entre l'utilisateur et le matériel (= abstraction des spécificités d'accès complexes du matériel)
- Le SE gère les ressources matérielles
- Le SE réalise 4 grands types de tâches :
  - La gestion des processus
  - La gestion de la mémoire
  - La gestion du système de fichiers
  - La gestion des périphériques d'E/S

Les concepts de base

- **Processus** :
  - programme en cours d'exécution auquel est associé du code, des données et des ressources

- **Mémoire** :
  - organe d'un ordinateur permettant d'enregistrer, de stocker et de restituer des données

- **Système de fichiers** :
  - structure de données permettant de stocker les informations et de les organiser dans des fichiers sur des mémoires secondaires (disque dur, disquette, CD-ROM, clé USB, disques SSD, etc.)

- **Périphériques d'E/S** :
  - composants de matériel informatique assurant les communications entre l'unité centrale de l'ordinateur et le monde extérieur, en particulier l'utilisateur (clavier, souris, scanner, webcam, imprimante, modem, etc.)

Complexité d'un système d'exploitation

- **Nombre d'utilisateurs de l'ordinateur** :
  - Mono-utilisateur
  - Multi-utilisateur

- **Nombre de processus à exécuter** :
  - Monotâche
  - Multitâche

⇨ **Partage des ressources** (CPU, mémoire, réseau, périphériques, etc.) entre processus et entre utilisateurs

Principaux systèmes d'exploitation aujourd'hui

Microsoft Windows
1.0 - 3.x - 95 - 98 - Me - NT - 2000 - XP - 2003 - Vista - Win 7, 8 , 10 et 11

GNU/Linux
Debian - Fedora - Gentoo - Mandriva - Red Hat - Slackware
SuSE - Ubuntu

Mac OS

Mac OS X -.0 -.1 -...- .15 - 11, 12

BSD
FreeBSD - NetBSD - OpenBSD - DragonFly BSD - PC-BSD

Autres
AmigaOS - BeOS - DOS - Inferno - LynxOS - Haiku - OS/2 - QNX - Solaris - UNIX
MVS - OS/360 - OS/390 - OS/400 - Plan 9 - ReactOS - VMS - ZETA - FreeDOS

Plan du cours

- Introduction aux systèmes d'exploitation
- **Présentation générale d'UNIX**
- Processus et parallélisme
- Ordonnancement
- Communication et synchronisation

Les origines

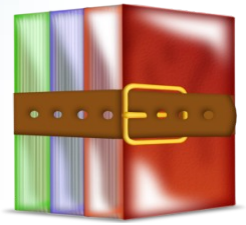
- UNIX est né au sein des laboratoires BELL (Filiale d'AT&T)
- Développé à partir de 1969 par Ken Thompson et Dennis Ritchie
- Dès 1973, UNIX est réécrit à 90% en langage C
- En 1975, les sources d'Unix sont diffusées dans les universités
- Développement de 2 branches :
  - **BSD** développé à l'Université de Berkeley (Californie)
  - **System V** vendu par AT&T à Sun Microsystems, IBM, DEC et HP
- **UNIX®** est une marque déposée depuis 1994

Caractéristiques principales

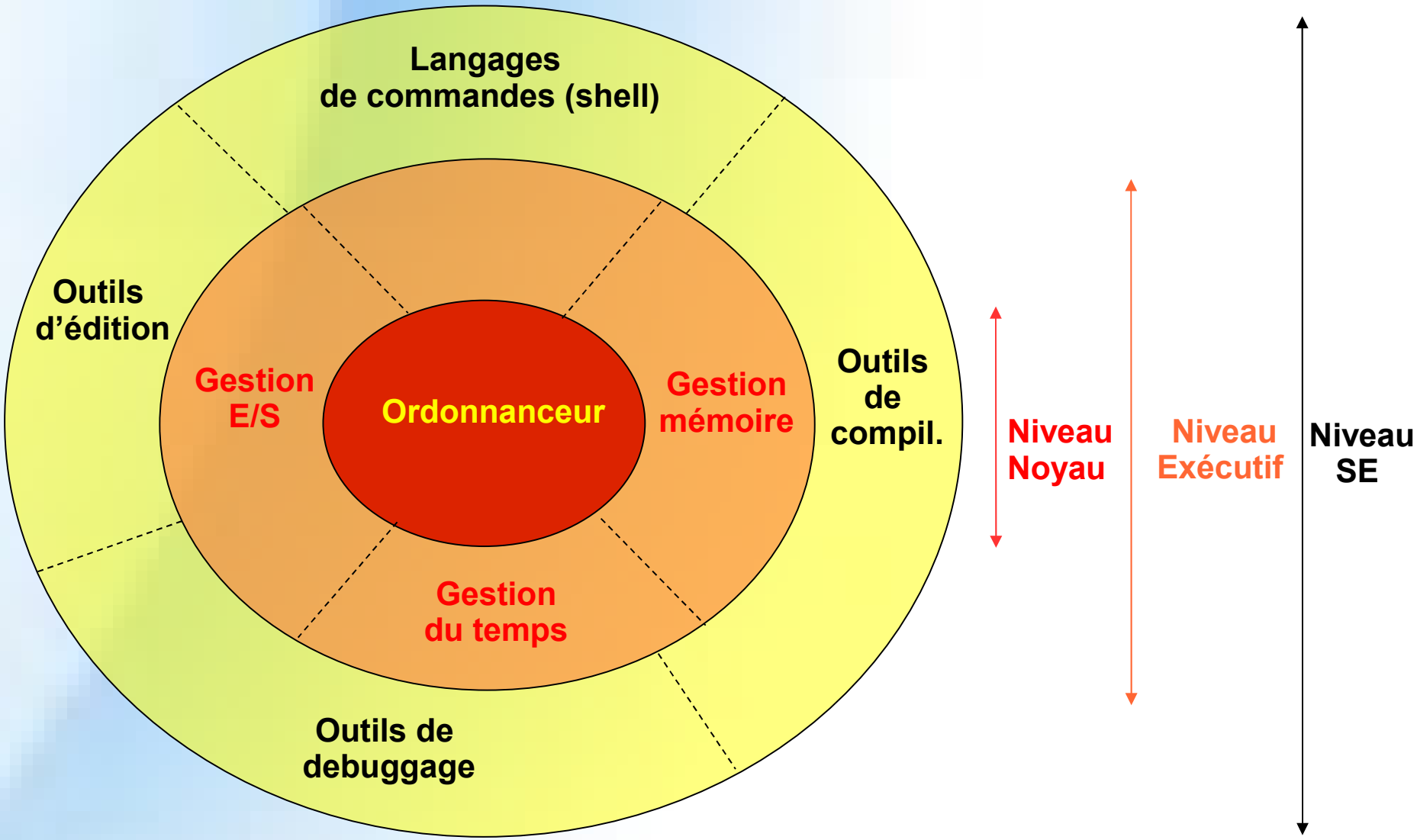
- Multitâche
- Multi-utilisateur
- Portable
- Interactif
- Système à mémoire virtuelle
- Un riche panel d'outils (plusieurs centaines)

Outils disponibles

- Editeurs de texte
- Développement logiciel
- Communication
- Documentation
- Bureautique



Structure générale



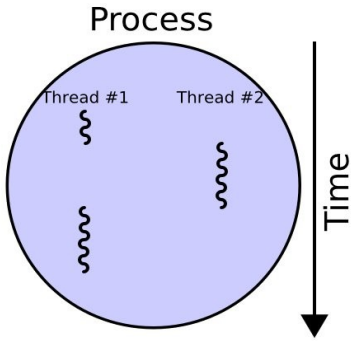
Plan du cours

- Introduction aux systèmes d'exploitation
- Présentation générale d'UNIX
- **Processus et parallélisme**
- Ordonnancement
- Communication et synchronisation

Processus et threads : définition (1)

- Un processus est un programme **en cours d'exécution**
  - Programme (= instructions machine) : entité statique
  - Processus (= réalisation d'actions) : entité dynamique

- A un **processus** sont associés plusieurs éléments :
  - son code
  - ses données
  - les ressources qui lui sont attribuées
  - un ou plusieurs threads d'exécution



Processus et threads : définition (2)

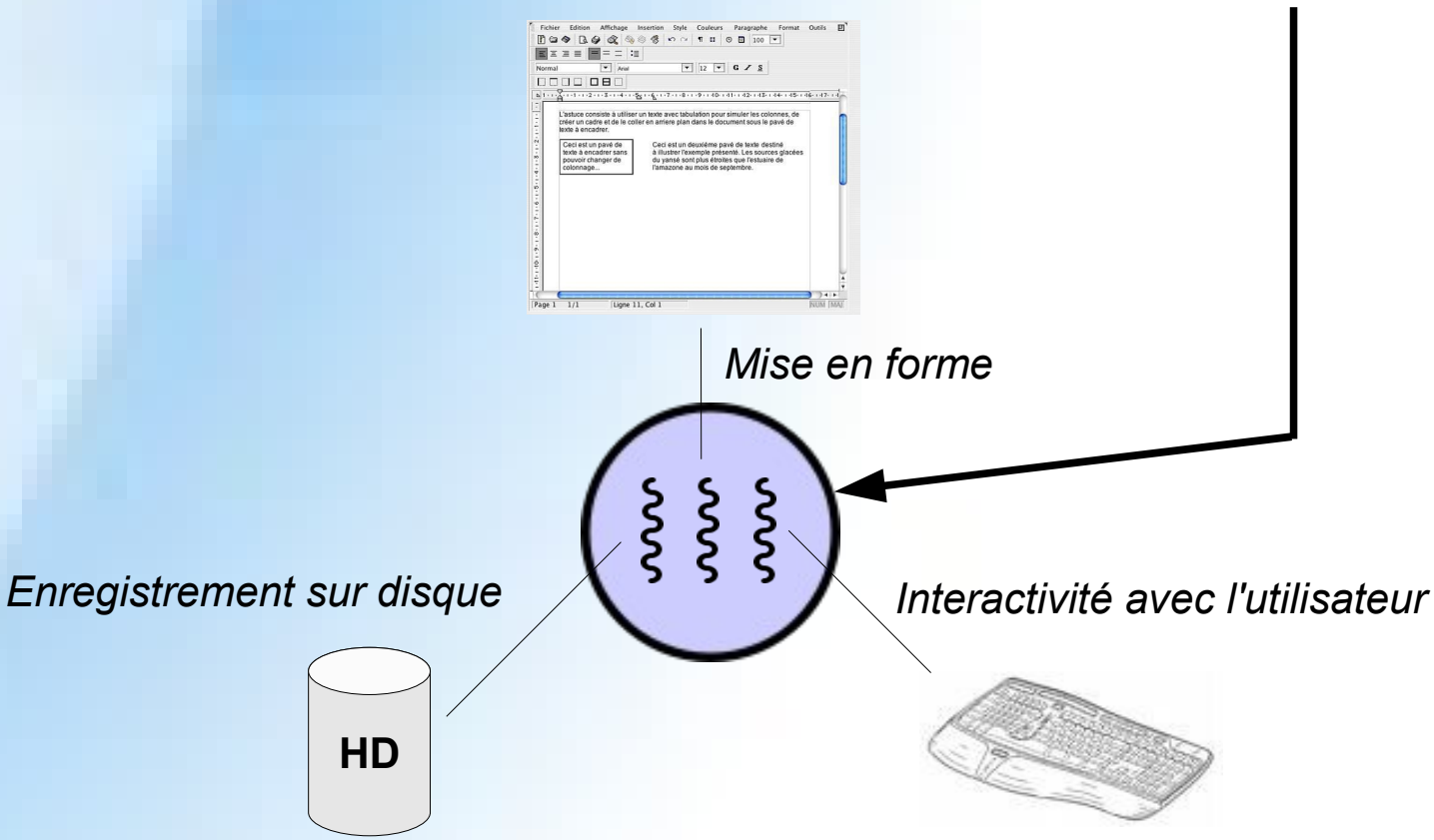
- Un **thread** est un flot d'exécution dans le code du processus doté :
  - d'un compteur programme (suivi des instructions à exécuter)
  - de registres systèmes (variables de travail en cours)
  - d'une pile (historique de l'exécution)

- Plusieurs processus permettent à un ordinateur d'effectuer plusieurs tâches à la fois. Ils se partagent les **ressources physiques**.

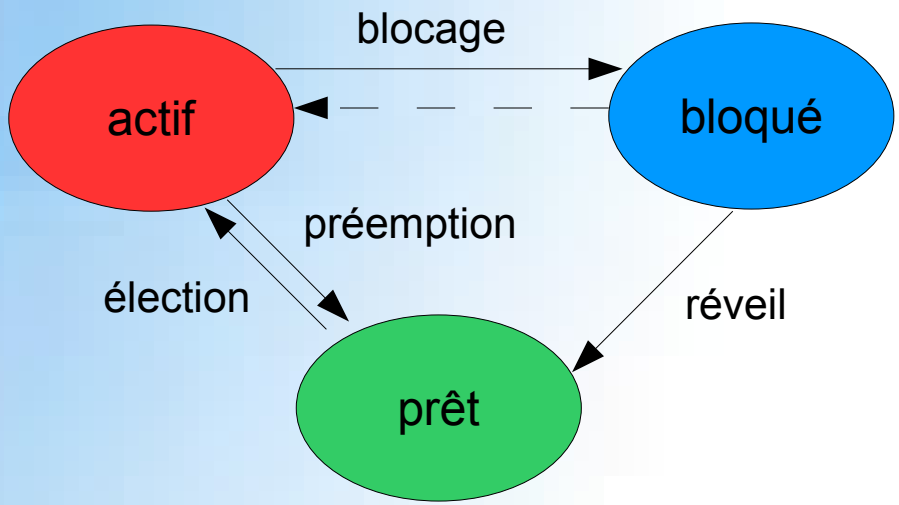
- Plusieurs threads permettent à un processus de décomposer le travail à exécuter en parallèle. Ils se partagent les **ressources physiques et virtuelles**.

Utilisation des threads

- Amélioration de l'interactivité d'un programme (ex : traitement de texte)



Etats d'un processus

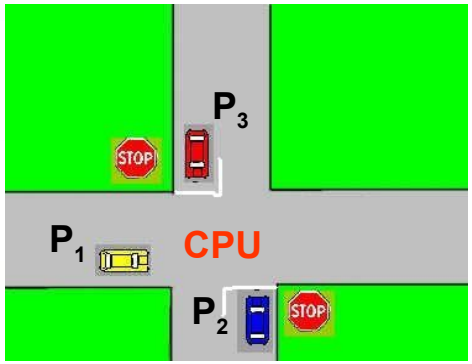


- actif : le processus s'exécute sur le processeur
- prêt : le processus est prêt à s'exécuter mais n'a pas le processeur
- bloqué : il manque au processus une ressource (en plus du processeur) pour qu'il puisse s'exécuter

Allocation du processeur aux processus

- Comment contrôler l'ordre de passage des processus sur le processeur ?
- Comment contrôler la répartition du temps d'exécution entre les processus ?

→ **L'ordonnanceur**



- L'ordonnanceur est un composant (procédure) du système d'exploitation

Plan du cours

- Introduction aux systèmes d'exploitation
- Présentation générale d'UNIX
- Processus et parallélisme
- **Ordonnancement**
- Communication et synchronisation

Ordonnancement : quels objectifs ? (1)

- L'ordonnancement consiste à **choisir** le processus à exécuter à un instant *t* et à **déterminer** le temps durant lequel le processeur lui sera alloué

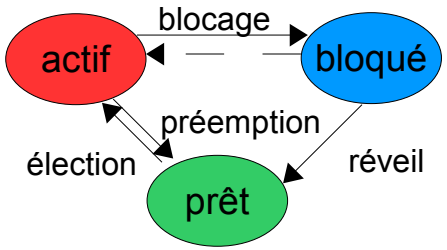
- L'**objectif de l'ordonnanceur** est d'optimiser certains aspects des performances du système.

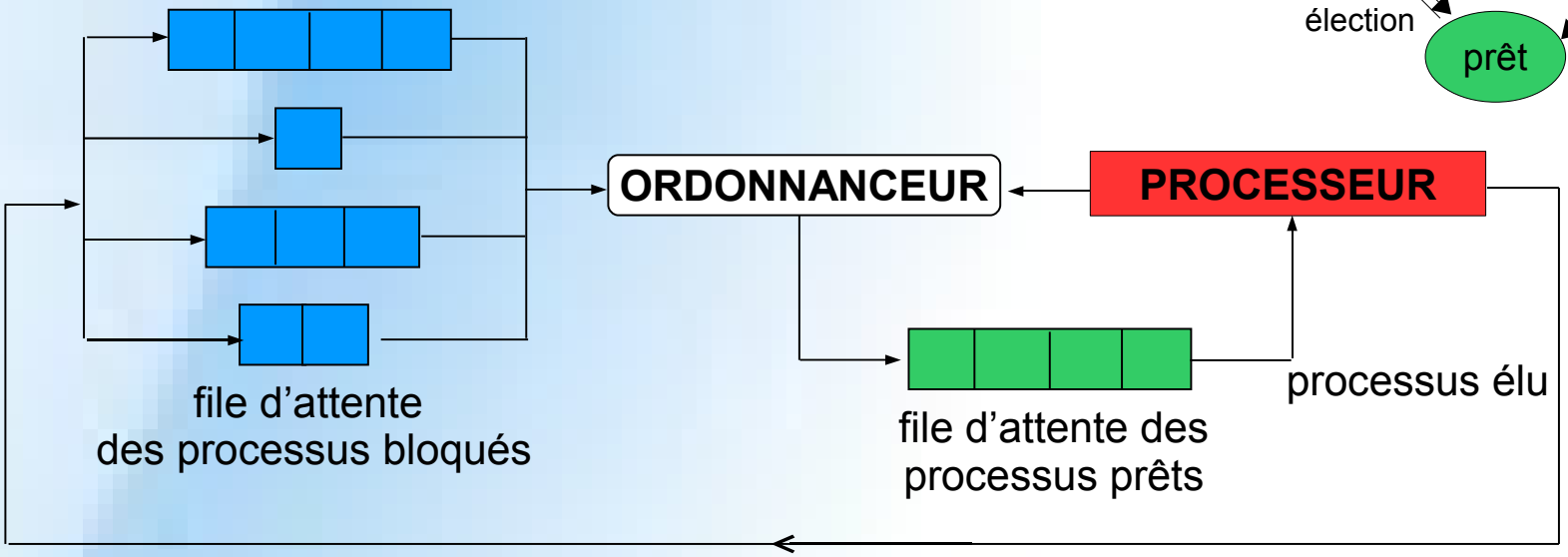
- **Compromis** entre :
  - Temps de traitement moyen du système
  - Utilisation efficace du processeur
  - Temps de réponse moyen/max du système
  - Satisfaction des conditions d'échéance pour les processus
  - Bonne utilisation des autres ressources du système

Ordonnancement : quels objectifs ? (2)

- **Systèmes de traitement par lots**
  - optimiser le nombre de processus à l'heure
  - réduire le délai entre la soumission et l'achèvement
  - faire en sorte que le processeur soit occupé en permanence
- **Systèmes interactifs**
  - répondre rapidement aux requêtes
  - répondre aux attentes des utilisateurs
- **Systèmes temps réel**
  - respecter les délais
  - eviter de perdre des données
  - eviter la dégradation de la qualité au niveau des résultats produits

Ordonnancement des processus

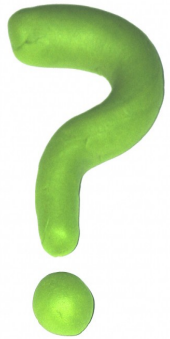




- **ORDONNANCEUR** : alloue le processeur aux différents processus selon un algorithme d'ordonnancement donné

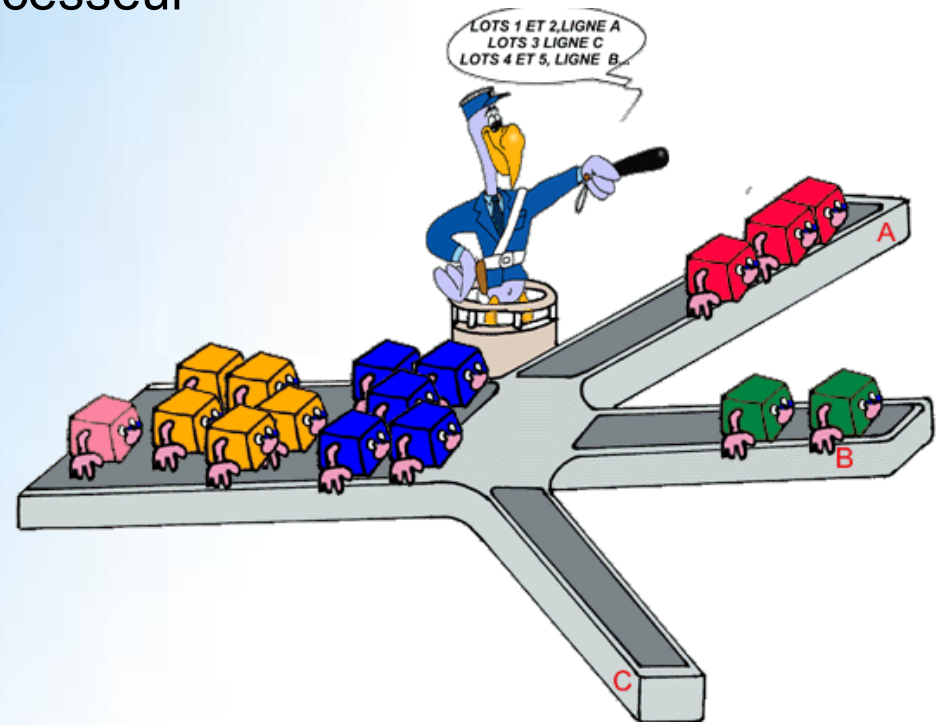
Critères d'ordonnancement

- Utilisation du processeur
- Débit (ou capacité de traitement)
- Temps de traitement
- Temps d'attente
- Temps de réponse
- Prévisibilité
- Equité
- Priorités



Typologie des algorithmes d'ordonnancement

- Monoprocesseur / multiprocesseur
- En-ligne / Hors-ligne
- Préemptif / Non préemptif
- Oisif / Non oisif
- Centralisé / Réparti



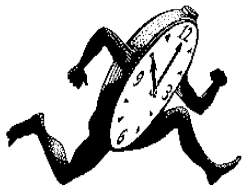
Ordonnancement non préemptif

- Ordonnancement **selon l'ordre d'arrivée** :
  - premier arrivé, premier servi

  First Come First Serve (FCFS)

- Ordonnancement **selon la durée de calcul** :
  - travail le plus court d'abord

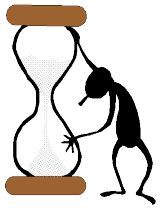
  Shortest Job First (SJF)



Ordonnancement préemptif

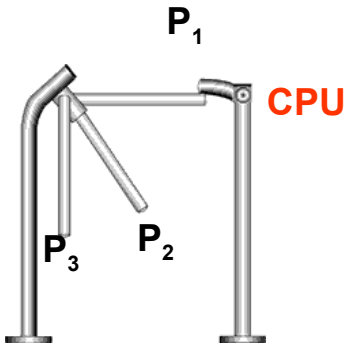
- Ordonnancement **selon la durée de calcul restante** :
  - temps restant le plus court d'abord

  Shortest Remaining Time (SRT)

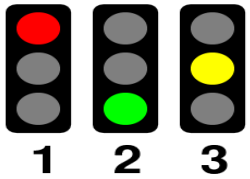


- Ordonnancement **sans notion de priorité** :
  - temps-partagé avec politique du tourniquet

  Round-Robin (RR)



- Ordonnancement **à priorités** (statiques ou dynamiques) :
  - la tâche la plus prioritaire obtient le processeur



Performance des algorithmes d'Ordonnancement

Un processus possède des paramètres d'entrée :
- Une date d'entrée ou d'arrivée
- Une durée d'exécution ou de traitement
- Une priorité

A prtir de ces donées, nous pouvons calculer les metriques suivants :

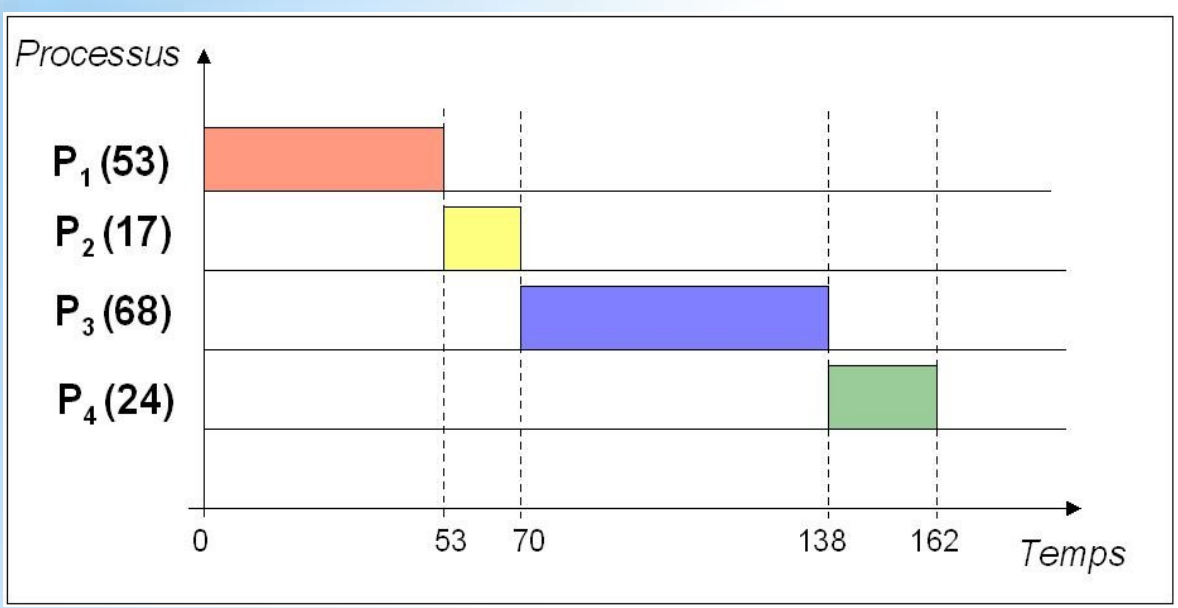
Temps de rotation = Temps fin d'exécution - Temps d'arrivée

Temps d'attente = Temps de rotation – Durée d'exécution

Rendement = Durée d'exécution / Temps de rotation

Ordonnancement First Come First Served (FCFS)

- **Principe** : Les processus sont ordonnancés selon leur ordre d'arrivée
- **Exemple** (P_1, P_2, P_3 et P_4 arrivent dans cet ordre à t = 0) :



Temps de rotation(P1)= 53
Temps de rotation(P2)= 70
Temps de rotation(P3)= 138
Temps de rotation(P4)= 162
Moyenne = 105.75

Temps d'attente(P1)= 0 ,
Temps d'attente(P2)= 53
Temps d'attente(P3)= 70
Temps d'attente(P4)= 138
Moyenne = 65.25

Rendement(P1)= 1
Rendement(P2)= 0.24
Rendement(P3)= 0.49
Rendement(P4)= 0.14
Moyenne = 0.46

Ordonnancement First Come First Served (FCFS)

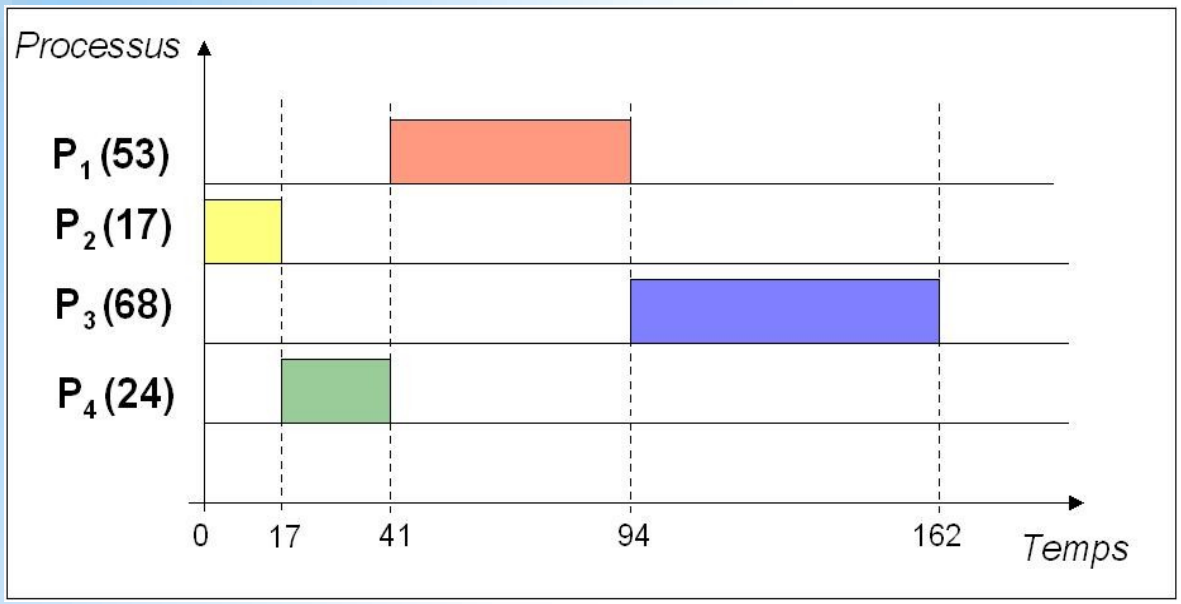
- **Intérêts** : 
  - Algorithme facile à comprendre
  - Faible complexité d'implémentation (une seule liste chaînée)

- **Inconvénients** : 
  - Pas de prise en compte de l’importance relative des processus
  - Temps d'attente du processeur généralement important

Ordonnancement Shortest Job First (SJF)

- **Principe** : Le processus dont le temps d'exécution est le plus court est ordonnancé en priorité

- **Exemple** (P_1, P_2, P_3 et P_4 arrivent à t = 0) :



Temps de rotation(P1)= 94
Temps de rotation(P2)= 17
Temps de rotation(P3)= 162
Temps de rotation(P4)= 41
Moyenne = 93.99

Temps d'attente(P1)= 41
Temps d'attente(P2)= 0
Temps d'attente(P3)= 94
Temps d'attente(P4)= 17
Moyenne = 38

Rendement(P1)= 0.56
Rendement(P2)= 1
Rendement(P3)= 0.41
Rendement(P4)= 0.58
Moyenne = 0.63

Ordonnancement Shortest Job First (SJF)

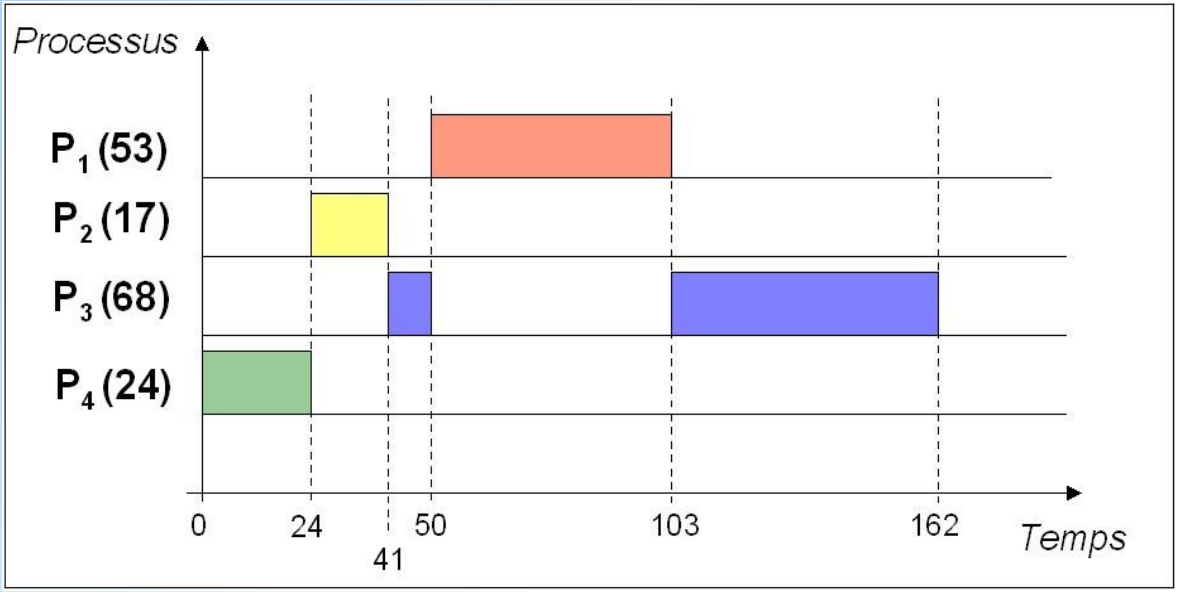
- **Intérêts** : 
  - SJF réduit le temps d'attente des processus
  - Utilisation limitée à des environnements et à des applications spécifiques

- **Inconvénients** : 
  - Pas de prise en compte de l'importance relative des processus
  - Algorithme optimal uniquement dans le cas où tous les processus sont disponibles simultanément

Ordonnancement Shortest Remaining Time (SRT)

- **Principe** : Le processus dont le temps d'exécution restant est le plus court parmi ceux qui restent à exécuter, est ordonnancé en premier

- **Exemple** (P_3 et P_4 arrivent à t = 0; P_2 à t = 20; P_1 à t = 50) :



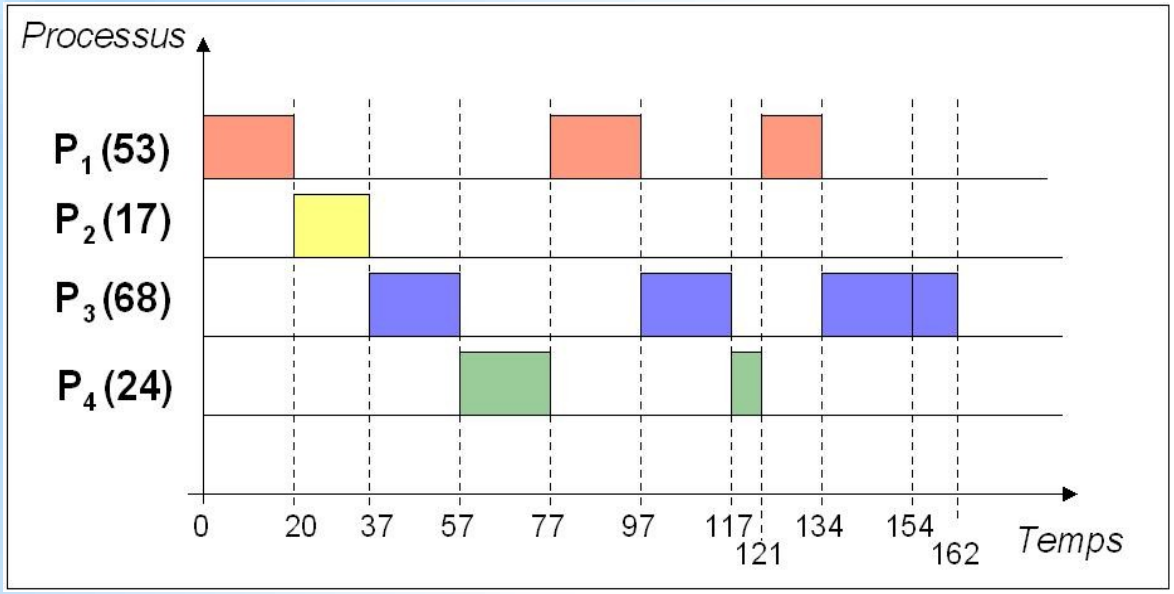
Ordonnancement Shortest Remaining Time (SRT)

- **Intérêts** : 
  - SRT minimise le temps d'attente moyen des processus les plus courts
  - Utilisation limitée à des environnements et à des applications spécifiques

- **Inconvénients** : 
  - Pas de prise en compte de l'importance relative des processus
  - Non équité de service : SRT pénalise les processus longs
  - Possibilité de famine pour les processus longs

Ordonnancement temps-partagé (Round-Robin)

- **Principe** : allocation du processeur par tranche (quantum) de temps
- **Exemple** (q=20, n=4) :



- Chaque tâche obtient le processeur au bout de **(n-1)*q** unités de temps au plus

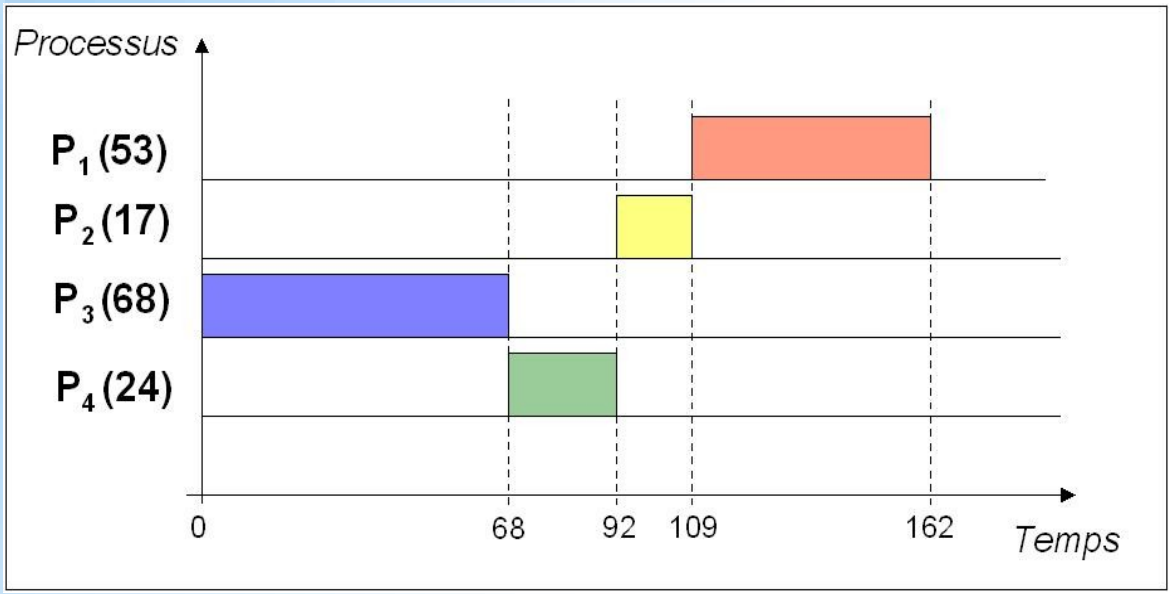
Ordonnancement temps-partagé (Round-Robin)

- **Intérêts** : 
  - Equité de l'attribution du processeur entre toutes les tâches
  - Mise en œuvre simple

- **Inconvénients** : 
  - Pas de prise en compte de l'importance relative des tâches
  - Difficulté du choix de la tranche de temps
    - Si q est trop grand, Round-Robin devient équivalent à FIFO
    - Si q est trop petit, il y a augmentation du nombre de changements de contexte !

Ordonnancement à priorités statiques

- **Principe** : allocation du processeur selon des priorités statiques (numéros affectés aux processus pour toute la vie de l'application)
- **Exemple** (*priorités*(P_1, P_2, P_3, P_4)=(3,2,0,1)) :



Dans certains systèmes, l'échelle des priorités est inversée (0 est alors la priorité la plus faible)

Ordonnancement à priorités statiques

- **Intérêts** : 
  - Prise en compte de l'importance relative des processus
  - Mise en œuvre simple
- **Inconvénients** : 
  - Problèmes de blocages durant des périodes de temps illimitées
  - Problèmes de famines des processus de moindres priorités