INP TOULOUSE ENSEEIHT

Systèmes d'exploitation pour les Télécoms/Réseaux

Introduction et rappels

Jérôme Ermont

Plan de la présentation

Au menu

1. Ce cours : quelques rappels sur les mécanismes de base des OS
2. Une introduction à la virtualisation
   - Un peu de docker
   - Un peu d'Android
   - TP initiation à Docker
3. Les micro-noyaux
   - Avant les L5 il y a L4
   - Un peu de seL4
4. Les OS pour l'IoT
   - Contiki / RIOT / FreeRTOS
   - mini-projet avec cartes TI Simplelink (ou Raspberry Pi)

Objectifs d'un système d'exploitation

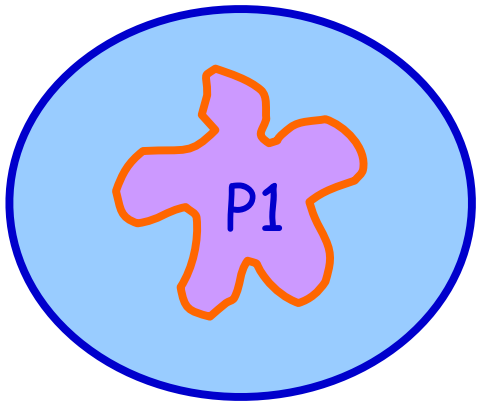
- Un système informatique
  - Stockage/traitement/transmission de l'information

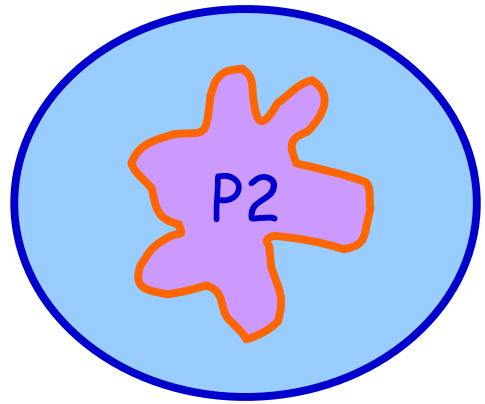
Un système d'exploitation, c'est quoi ?

- Une machine virtuelle
  - simple
  - pérenne
- Un gestionnaire de ressources
  - fiable
  - équitable

Notion de machine virtuelle

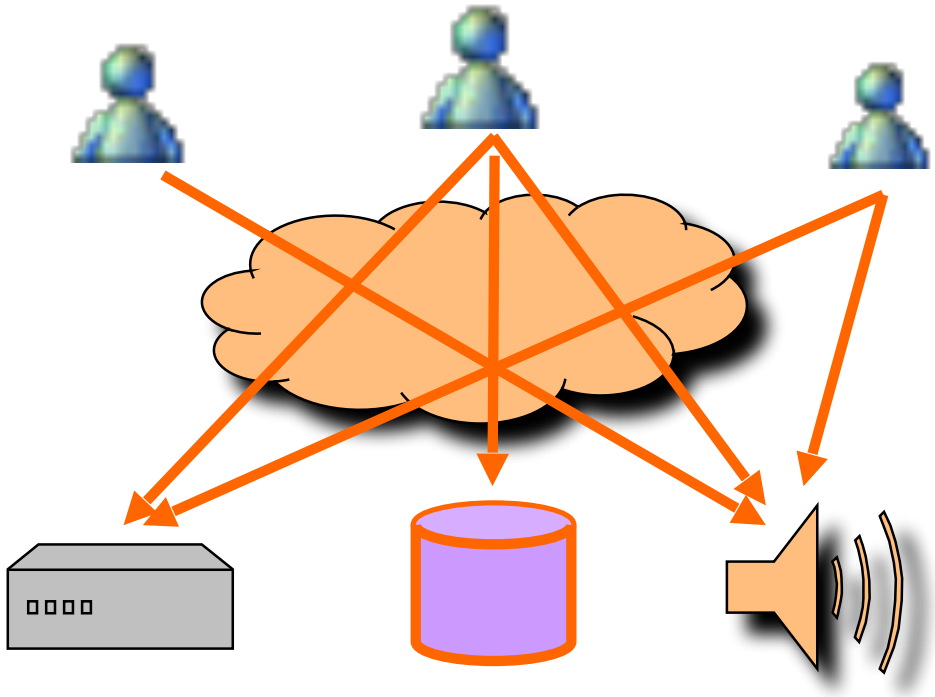
- Objectif : interface plus abstraite
  - plus pérenne
  - plus simple
  - plus homogène





Notion de gestionnaire de ressources

- Objectif : accès cohérents
  - fiabilité
  - stabilité
  - équité



Un système d'exploitation, c'est quoi ?

- Du logiciel :
  - Assembleur,
  - C,
  - Java,
  - ...
- Aidé par le matériel :
  - gestion mémoire
  - protection des accès
  - ...

Les éléments d'un système d'exploitation

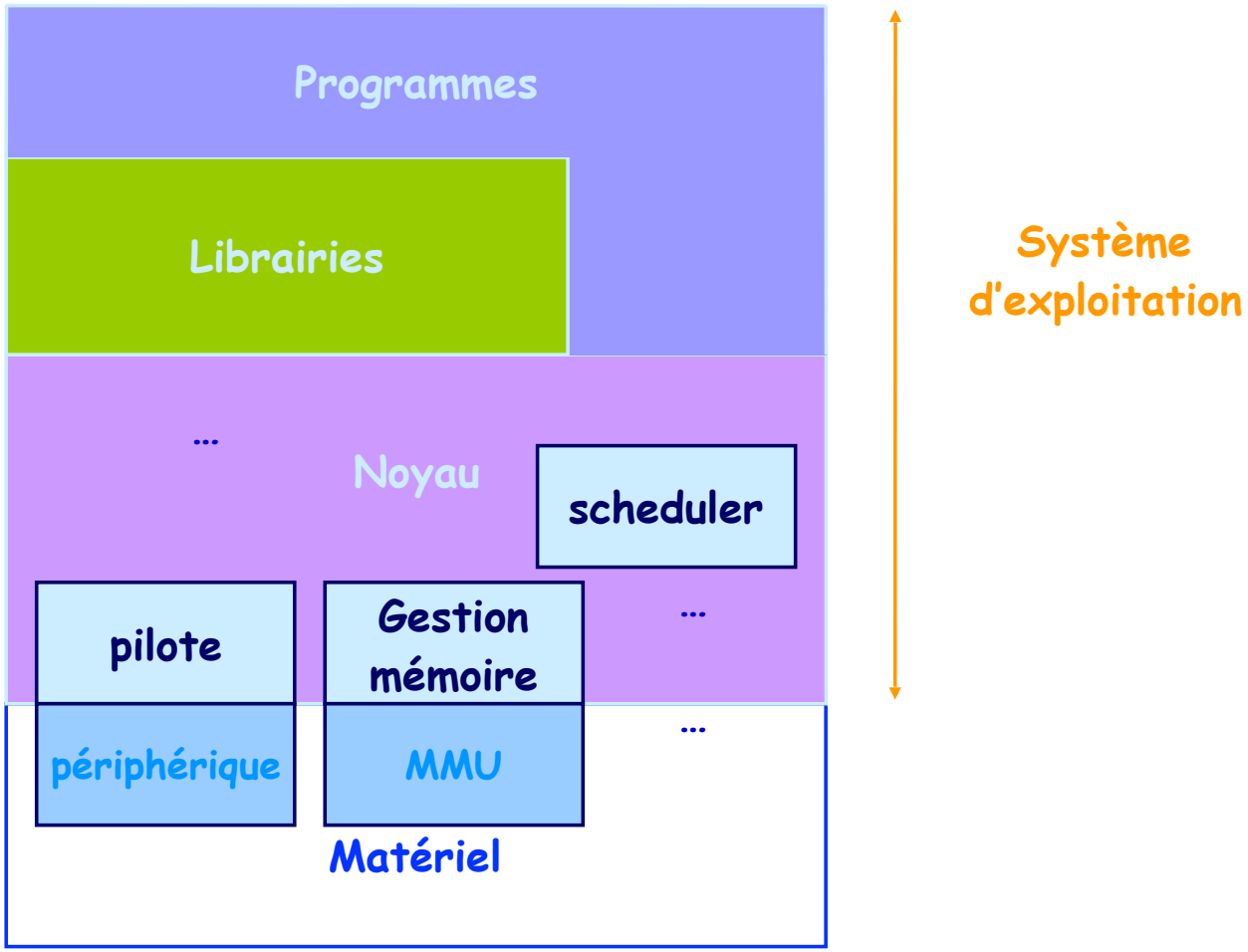
Noyau

- Briques logicielles de base permettant l'exécution de logiciels
- par ex. : Linux, Mach et dérivés (Hurd, XNU)

Systèmes d'exploitation

- Ensembles de logiciels
- par ex. : Debian, Android, MacOS

Architecture d'un système d'exploitation



Les abstractions

Machine virtuelle → abstractions

- Définition d'objets plus abstraits
- Permettant d'assurer un service équivalent exempt de contraintes physiques

Stockage et traitement des données pour des utilisateurs

- Utilisateur humain (identifiants)
- Stockage (système de fichiers)
- Exécution d'un traitement (processus, tâche)

Notion de tâches

- Abstraction de l'éxecution d'un programme
- Processus, tâche, thread, ...
  - Différentes gestions
- Exécution d'une suite d'instructions
  - Eventuellement discontinue dans le temps
- Le noyau n'est pas une tâche
  - En général
  - Certains processus aident le noyau

Notion de parallèlisme

- Abstraction d'un lot de calculateurs
  - Généralement dynamique
- Exécution simultanée de plusieurs tâches
  - Plusieurs organes d'exécution
- Pseudo-parallélisme
  - Fournir plus de calculateurs virtuels que de calculateurs physiques
- Systèmes mutli-tâches
  - OS généralistes actuels

Notion de système de fichiers

- Abstraction des supports
  - Stockage à long terme des données
  - Outil de base de sauvegarde de l'information
- Généralement agnostique à la structure interne
  - Flux d'octets
  - Accès aléatoires
- Doté d'information de gestion
  - Informations associées
    - Nom symbolique, propriétaire, ...
  - Mécanismes de protection
    - Eviter les erreurs/malversations

Définition d'une interface

- Comment utiliser une machine (virtuelle) ?
  - Au travers d'une interface
- Comment solliciter un gestionnaire de ressources ?
  - Au travers de mécanismes d'accès
- Si maîtrise du système complet
  - Simples conventions d'utilisation (MV)
  - Personne ne risque de contourner l'interface
- Sinon
  - L'interface doit être un passage obligatoire
  - Techniques de protection
    - Ex. : Appels système

Classification des systèmes

- Systèmes réels nombreux et variés
  - Du lecteur mp3 au supercalculateurs ...

Différents types de systèmes

- Systèmes généralistes
  - Ordinateurs grand public, certains équipements réseau, téléphones portables, ...
- Systèmes spécialisés
  - Electroménager, équipements réseau, calculateurs embarqués, ...

Quelques caractéristiques marquantes

- Système temps réel
  - Contraintes de temps impérieuses
- Système parallèle
- Système réparti
  - englobe plusieurs systèmes physiques
- Système virtuel
  - JVM, émulateurs (MAME

Classification des systèmes

- Systèmes propriétaires
  - Code source appartenant à un éditeur
  - Ex. : Microsoft Windows, Apple MacOS, certains Unix, ...
- Systèmes « ouverts »
  - Fondés sur du code accessible
  - Ex. : machine Java, certains Unix (Linux, *BSD)
- Normes POSIX
  - Assurant une certaine interopérabilité (source)

Exemples connus

- Linux
  - Noyau « Unix-like »
  - Associé à l'environnement GNU
  - Assez largement conforme à POSIX
- Machine virtuelle Java
  - Sur un système hôte
  - Sur un système « réel »

Programmation système

- Programmation d'applications proches du noyau
- Typiquement : applicatifs permettant de gérer le système
- Programmation du système
  - écriture de pilotes
  - ajout/évolution de fonctionnalités

Plan de la présentation

Qu'est-ce qu'une tâche ?

Du point de vue de l'utilisateur

- Abstraction de l'exécution d'un programme sur un système réel
  - Exécution (voir interprétation) sur un système virtuel
  - Code écrit dans un langage évolué (puis éventuellement compilé)
- Aussi indépendant des autres que possible
  - Qualité de la machine virtuelle

Qu'est-ce qu'une tâche ?

Du point de vue du noyau

- Une entité à gérer
  - Création, contrôle, destruction
- Elément de dynamique
  - Exécution de code
  - Sollicitation/libération de ressources
- Caractéristiques
  - Etat processeur
  - Ressources utilisées
  - Propriétés au cœur de l'OS

Ordonnancement

- Cohabitation temporelle des tâches
- Mise en œuvre
  - Assujettie à des contraintes variées
  - Multiples solutions
- Réalisé par l'ordonnanceur
  - Elément fondamental de l'OS
  - Aidé par le matériel

Ordonnancement

Principe de base de l'ordonnanceur

- Un certain nombre de tâches sont éligibles (prêtes à être exécutées)
- Si aucune tâche n'est en cours d'exécution, l'ordonnanceur choisit une tâche qui éligible qui est alors exécutée
- Lorsque la tâche en cours d'exécution se termine, l'ordonnanceur choisit une nouvelle tâche

Cohabitation des tâches

- Comment exécuter plusieurs tâches sur un seul processeur ?
- Monotâche

- Multitâche (pseudo parallélisme)

Le multitâche : pourquoi ?

- Partager équitablement le temps processeur
  - Notion de quantum
- Structurer les traitements
  - Si intéractions entre deux traitements
- Pour utiliser au mieux le processeur
  - Inactivité en cas d'entrée/sortie
  - Lenteur de l'environnement

Partager équitablement le processeur

Objectif

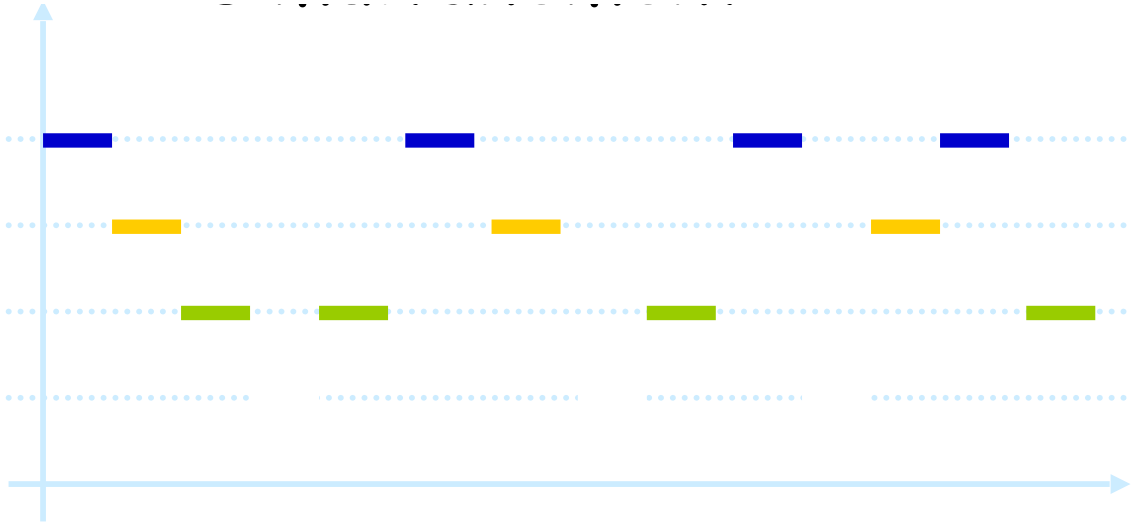
- Permettre à n tâches d'avancer de façon équitable

- Chaque tâche dispose d'un processeur virtuel
  - Somme des puissances inférieure à la puissance réelle
- Sur un processeur physique
  - A tout moment, une seule tâche s'exécute réellement

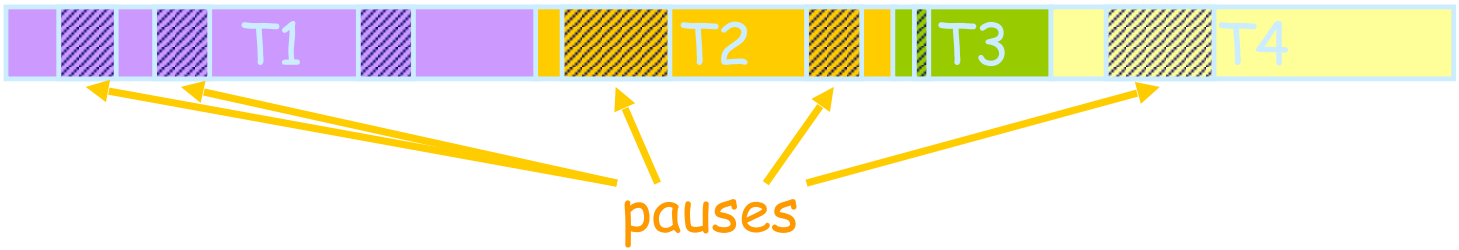
Partager équitablement le processeur

- Technique : définition d'un quantum
  - Durée maximale d'exécution continue

- Si le quantum est suffisamment petit, les tâches semblent évoluer « simultanément »



Utiliser au mieux le processeur



- Les tâches n'exécutent pas toujours du code
  - Attente de ressources
  - Attente d'événements

- Le pseudo-parallélisme permet de mettre à profit ce temps « perdu »

Utiliser au mieux le processeur

- Ne pas laisser un processeur inactif si une tâche est éligible
- Si la tâche en cours doit attendre
  - Elle n'est pas éligible
  - Une nouvelle tâche est choisie
  - La tâche devra redevenir éligible
- Basculement avant la fin du temps imparti (Quantum)

Pourquoi le multitâche ?

Le cas des OS généralistes

- Objectifs
  - Utiliser au mieux la puissance de calcul
  - Optimiser l'interactivité
- Contraintes
  - performances
  - rarement strictes
  - pas de métrique

Pourquoi le multitâche ?

Optimiser l'interactivité

- Favoriser les applications interactives
  - tâche réalisant des E/S
  - pénalisant pour le calcul
- Nécessité d'une priorité
  - statique
  - dynamique

Ordonnancement

- A l'initiative de qui ?
- Quand ?
- Comment ?
- Quand choisir la tâche ?
- Vers quelle tâche ?
- Quelle vision de la mémoire ?

A l'initiative de qui ?

- De la tâche en cours d'exécution
  - Système collaboratif
  - par ex. : Windows 3.1
- De l'ordonnanceur
  - Système préemptif
  - La plupart des OS modernes

Quand ?

- Lorsque la tâche le décide
  - Collaboratif
- Lorsque la tâche est bloquée
  - Attente d'E/S
  - Attente d'une ressource
- Lorsque la tâche a consommé son quantum
  - Préemptif

Exemple : invocation de l'ordonnanceur Unix classique

- Fin du quantum
  - Favoriser la dynamique
- Attente ressource
  - initiative du noyau
  - appel système
- Explicite
  - rare

Comment ?

Invoquer l'ordonnanceur

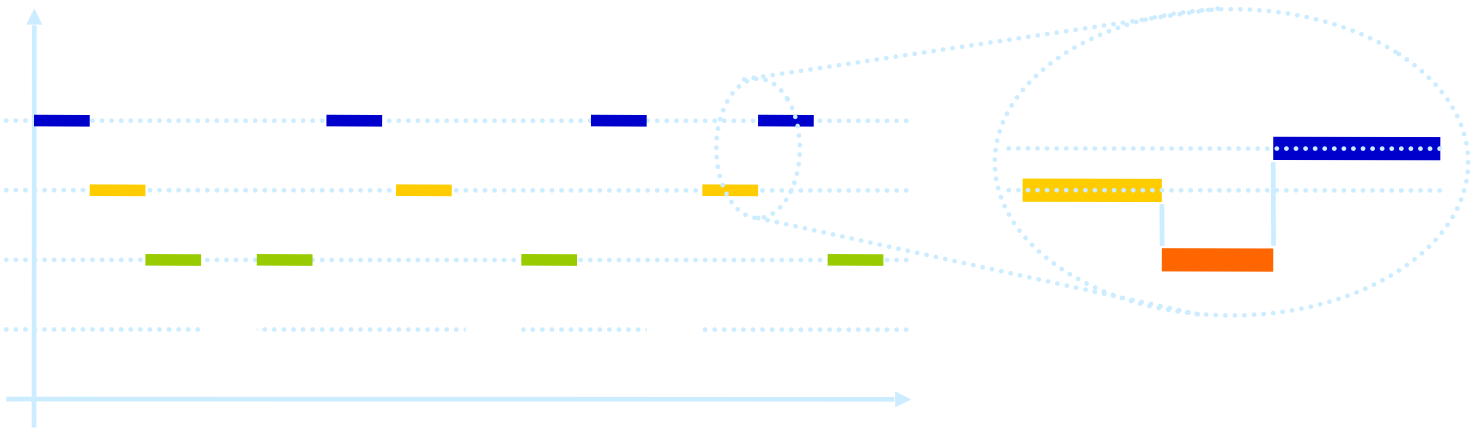
- Collaboratif
  - Appel explicite au noyau
- Préemptif
  - Préemption lors d'un appel au noyau
  - Interruption d'horloge

Basculer vers la nouvelle tâche

- Sauver l'état du processeur
  - registres $\rightarrow$ structure
- Restaurer l'état de la tâche cible
  - structure $\rightarrow$ registres

Comment ?

- Temps de basculement
  - Changement d'état + décision
  - Influence du quantum



Quand prendre la décision ?

Au dernier moment (on-line)

- permet de tenir compte de l'état
- accroissement du temps de basculement

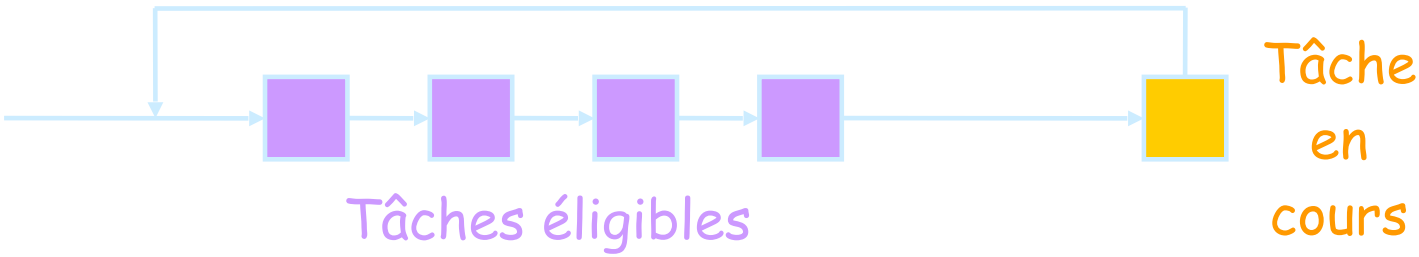
Avant toute exécution (off-line)

- nécessite la connaissance de profils temporels
- parfait pour les tâches périodiques (temps réel)

Quelle tâche ?

- Algorithme fondamental de l'ordonnacement
  - Choisir la tâche qui doit être exécutée
- Algorithmes monotâches
  - Premier arrivé premier servi
  - ...
- Algorithmes multitâches généralistes
  - Round-robin
  - Priorité (statique et/ou dynamique)
  - ...
- Algorithmes temps réel
  - Périodique off-line
  - ...

Le tourniquet ou round-robin



- Extension multi-tâche du PAPS
  - Tâches dans une file
  - Exécution chacun son tour
  - Cyclique (« round-robin »)

- Très simple et équitable

Plus court d'abord

- Idée : minimiser le temps d'attente moyen
  - Retarder un peu celui qui doit durer longtemps
- Nécessite une connaissance de la durée
  - Possible sur le long terme
  - Estimation sur court terme

Priorité

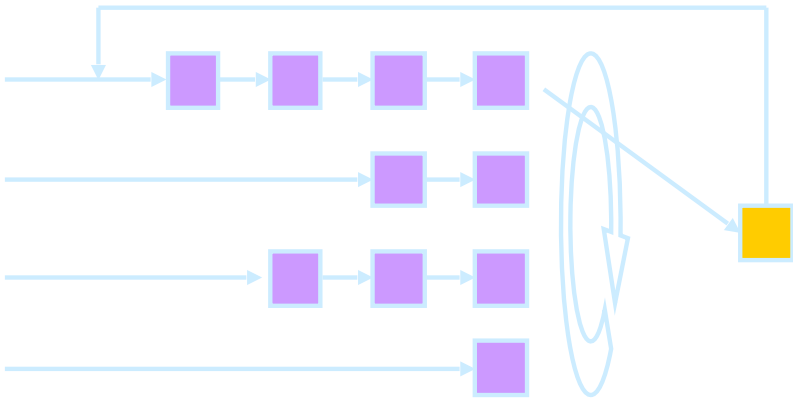
- Plus prioritaire d'abord
- Si arrivée plus prioritaire
  - préemption
  - risque de blocage
    - ressource bloquée par moins prioritaire
    - processeur bloqué par plus prioritaire
- Risque de famine
  - moins prioritaire

Quelle priorité ?

- Fixe
  - round-robin sur priorité max
  - risque de famine ailleurs
- Dynamique
  - quelle politique d'évolution ?
  - quelles garanties ?
- Compromis

Nombre limitée de priorités

- Gestion multi-niveau
  - Round-robin à priorité donnée
  - Une file par niveau de priorité
- Risques de famine



Ordonnancement périodique

- Systèmes spécifiques
  - Tâches identifiées
  - Temps réel
- Ordonnancement déterminisme
  - Prouver qu'il est possible dans les temps voulus
  - Offline

Ex : ordonnancement Unix classique

Priorité

- Base statique
- Composante dynamique
  - croît avec l'utilisation du processeur
  - favorise l'interactivité
  - empêche la famine

Tourniquet

- A priorité constante

Ordonnancement et mémoire

Quel partage de mémoire entre les machines virtuelles qui hébergent les processus ?

- Mémoires disjointes
- Mémoire commune

- Impact sur les performances
  - Mémoires disjointes : lourd !

- Conséquences sur la virtualité
  - Mémoire commune : pas de vrai système virtuel

Gestion à deux niveaux

- Processus lourds
  - espaces mémoires disjoints
  - mise en œuvre dans le noyau
- Processus légers
  - cohabitent au cœur d'un processus lourd
  - mise en œuvre dans
    - l'application
    - le noyau

Gestions indépendantes

- Niveau application
  - gestion des processus légers
  - basculement économique
- Niveau noyau
  - gestion des processus lourds
  - basculement via noyau
- Bilan
  - l'application ne tire pas parti des processeurs

Gestion unique ?

- Par le noyau
  - Coûteux
  - Uniforme
  - (par ex. Linux)
- Pas de connaissance de l'application
  - Distribution des tâches sur les processeurs ?

Gestions collaboratives

- Notion de processeurs virtuels
  - ordonnancés sur processeur(s)
  - créés dynamiquement
- Processus légers
  - ordonnancés sur processeurs virtuels
- Lourd

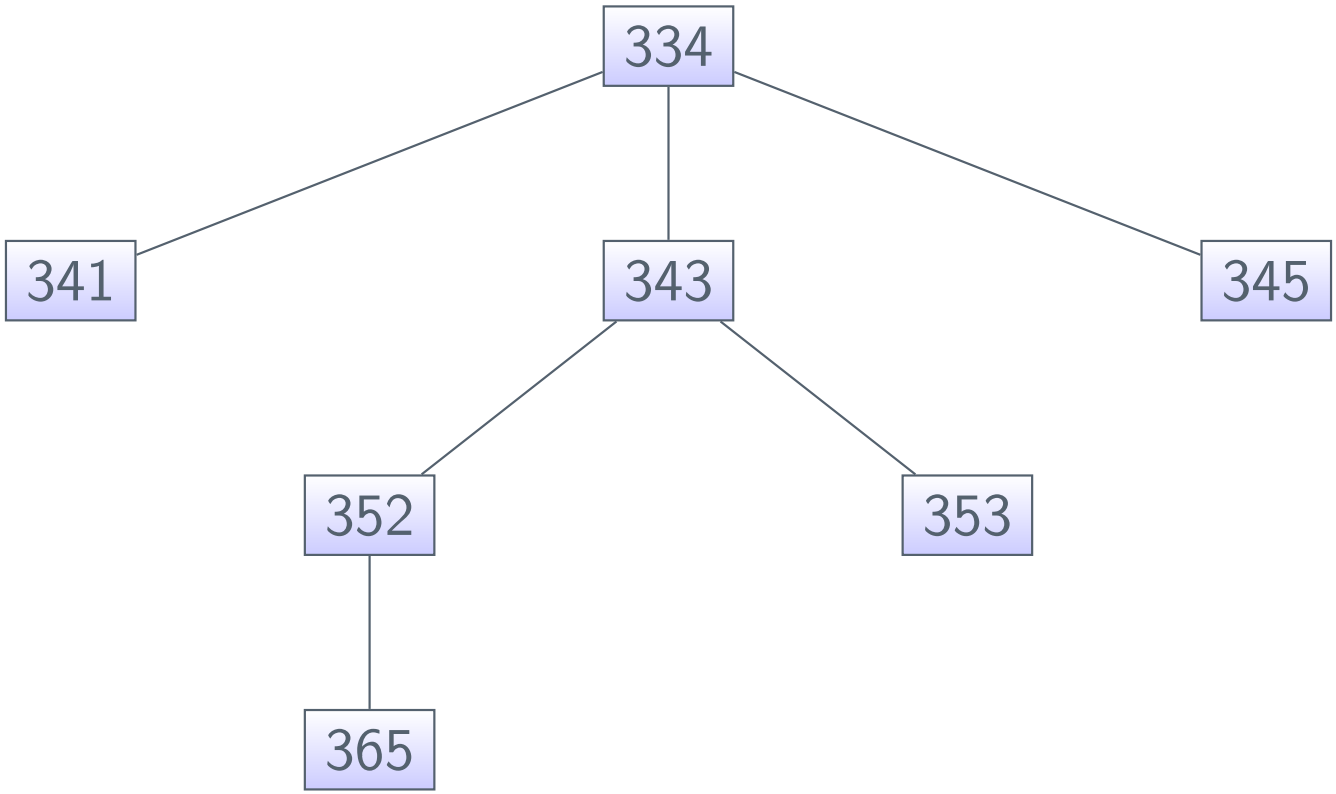
Mise en œuvre

- Gestion des processus
  - priorité
  - état du processus
  - état du processeur
- Gestion des listes
  - processus prêts
  - processus en attente

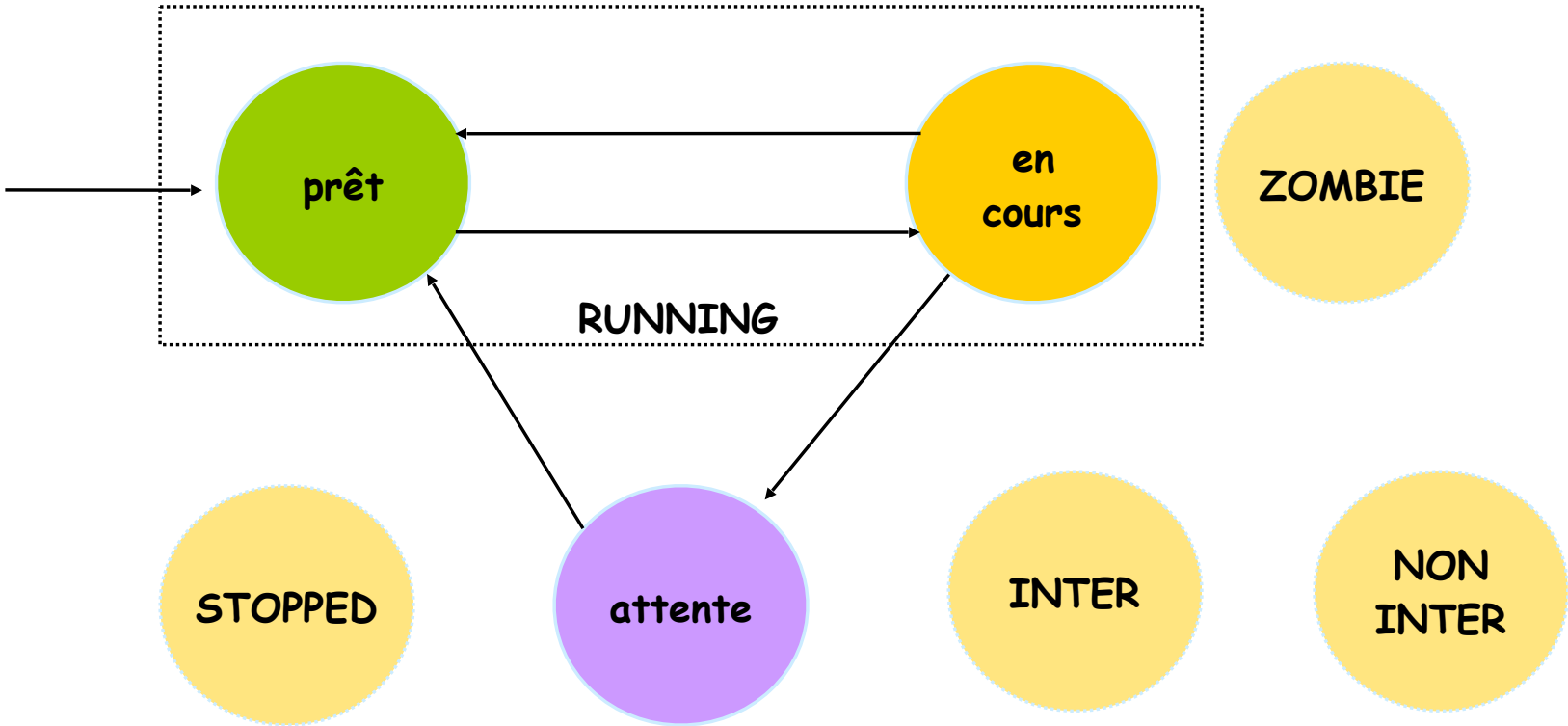
Caractéristique de base sous Unix

- Identifiant : PID
- Propriétaire
- Priorité
- Gestion de la mémoire
  - espace d'adresses distincts
- Gestion des fichiers
- ...

Arborescence



Etats d'un processus



Un exemple : Linux 2.6

- Temps constant : O(1)
- Favorise l'interactivité
- Equitable
- Bon comportement quelque soit le nombre de tâches
- Bonne gestion du SMP

Autres mécanismes

- Partage de charge entre les processeurs
  - si processeur inactif
  - régulièrement
- Prise en compte de l'interactivité
  - incrément de `sleep_avg` après attente
  - décrément lors de l'exécution
  - calcul du quantum

Principes

- Une file d'attente par processeur
- File d'attente « double »
  - processus actifs
  - processus expirés
- Une liste par priorité
  - bitmap identifiant liste non vide
  - maj des caractéristiques de basculement