ASR2 Système

Stéphanie Moreaud

Département d'informatique
IUT Bordeaux 1

Plan

Plan

Objectifs

Comprendre les principes de fonctionnement des systèmes d'exploitation multitâches et multi-utilisateurs, au niveau :

- de l'utilisation
- de la structure interne
- de la mise en œuvre

Organisation du module

4 cours de 1h50
13 séances de TDs de 1h50

Évaluation :

- contrôle continu : une note de TD
- 2 notes de DS

Plan du cours

- Introduction aux systèmes d'exploitation, notion de processus et ordonnancement
- Gestion de la mémoire
- Stockage des données et système de fichiers.
- Virtualisation, tendances...

Bibliographie

 Tanenbaum (A.), *Systèmes d'exploitation*. Pearson Education.

 Billaud (M.), *Cours de ASR2-Système*.

`http://www.labri.fr/perso/billaud/`.

 Courtès (L.), *Introduction aux systèmes d'exploitation*.

`http://people.bordeaux.inria.fr/lcourtes/`.

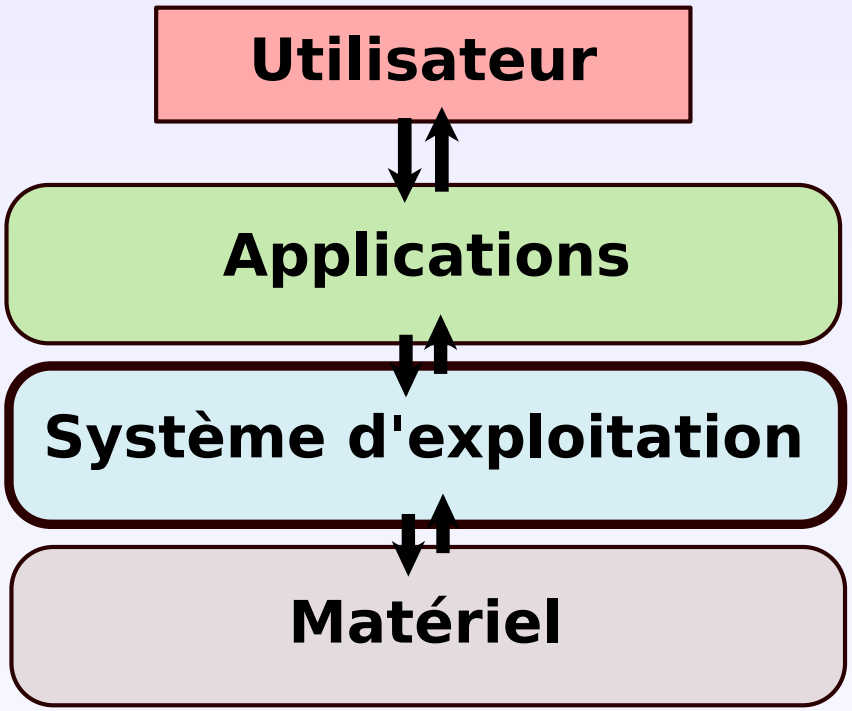
Remerciements à Michel Billaud et Ludovic Courtès

Plan

Qu'est ce qu'un système d'exploitation ?

Définition :un Système d'Exploitation (*Operating System*) est une couche d'abstraction (ensemble de programmes) construite au dessus du matériel pour

- masquer la complexité du matériel
- arbitrer l'accès aux ressources



Rôle d'un système d'exploitation

Le système d'exploitation doit permettre :

- d'exploiter les ressources matérielles efficacement
- de fournir aux programmeurs d'applications un environnement (machine étendue ou virtuelle) plus simple à programmer que la *machine réelle*.

Il est notamment :

- support d'abstraction du matériel
- piller de la gestion des ressources
- garant de la sûreté de fonctionnement (sécurité, tolérance aux fautes, gestion des erreurs, etc.)

Rôle d'un système d'exploitation

Le système d'exploitation cache les détails matériels sous une couche d'abstraction.

- communication réseau par différents moyens,
- utilisation de fichiers sur différents supports, ...

Il gère l'utilisation des ressources

- processeurs, mémoire, périphériques, ...

Il veille à ce qu'un certains nombre d'*opérations dangereuses* ne puissent être effectuées qu'avec des droits privilégiés.

Plan

Évolution des systèmes

Première génération (1945-1955) : machines à tubes.

- manipulation d'interrupteurs à bascule
- début des années 50, introduction des cartes perforées
- une seule tâche à la fois, intervention de l'opérateur avant/après chaque travail

  ➔ système monotâche

Seconde génération (1955-1965) : machines à transistors

- création des systèmes de *batch* (traitement par lots)
- enchaînement automatique d'un *train de travaux* assuré par un programme spécial (ancêtre des SE)

  ➔ système à traitement par lots

Évolution des systèmes

Troisième génération (1965-1971) : circuits intégrés

- exécution de plusieurs programmes chargés en mémoire
- réponse au blocage sur les périphériques

➔ premiers systèmes d'exploitation (OS/360, 1966), multiprogrammation

- réponse à la demande d'interactivité des utilisateurs

➔ généralisation des systèmes à temps partagé

- introduit en 1962 par le CTSS (*Compatible Time Sharing System*) du MIT

Évolution des systèmes

- utilisation du principe de multiprogrammation avec un passage d'un programme à l'autre après quelques millisecondes

➔ système multitâche

  - illusion que les programmes s'exécutent simultanément
  - Multics (MIT, 1965), premier système multitâche multi-utilisateurs
  - UNIX créé en 1970 sur les bases de Multics, facile à porter

Évolution des systèmes

Quatrième génération (1972-aujourd'hui) : micro-ordinateurs

- création de systèmes d'exploitation adaptés aux premiers micro-ordinateurs
  - CP/M, QDOS, MS-DOS
- intégration bonus des premières interfaces graphiques
  - Macintosh *System* (1984),
  - environnement Windows sur MS-DOS
  - etc.

Bilan : types de systèmes

L'évolution du matériel et des besoins a donné lieu à différents types de systèmes :

- monotâche (ou *monoprogrammation*) : un programme à la fois, intervention d'un opérateur entre chaque traitement
- monotâche à traitement par lots : enchaînement automatique d'un *train de travaux* assuré par un programme.
- multiprogrammation : plusieurs programmes en mémoire, profitent des temps morts pour s'exécuter.
- multitâche, temps-partagé : plusieurs programmes en mémoire exécutés quantum par quantum, donne l'illusion de l'exécution simultanée

Qu'en est-il aujourd'hui ?

Interconnexion d'ordinateurs (réseaux)

Poursuite de la miniaturisation

- généralisation du multicœur

➔ intégration de plusieurs processeurs sur une même puce

Multiplications des ressources

- cœurs, périphériques, etc.

Systèmes embarqués (téléphones, agendas, ...)

- fortes contraintes matérielles

➔ Complexification des aspects gérés par les systèmes d'exploitation

➔ Différents types de systèmes d'exploitation pour répondre à des besoins précis

- serveurs, multiprocesseurs, personnels, distribués, temps réel, embarqués, etc.

Plan

Composition du système d'exploitation

Les systèmes d'exploitations actuels comprennent généralement :

- un noyau
  - ➔ proche du matériel
  - ➔ modulaire sur beaucoup de systèmes
- des bibliothèques
  - ➔ code factorisé entre les applications
- des outils
  - ➔ ensemble de programmes et de scripts

Il s'attache à la gestion de plusieurs éléments :

- processus, mémoire, périphériques, fichiers, droits, informations, ...

Processus

Le système assure le bon fonctionnement des processus qui utilisent du temps et de la mémoire

- partage de l'espace mémoire disponible
- répartition du temps de fonctionnement
- éventuelle attribution des processeurs
- partage/protection de la mémoire entre processus
- terminaison de processus défaillant.

Droits

Lors de leur enregistrement, les utilisateurs se voient accorder des droits d'accès limités, de même que les applications utilisateurs.

Le système d'exploitation fait respecter les limitations imposées.

- garantit que les ressources ne sont utilisées que par les programmes et utilisateurs possédant les droits adéquats.

Périphériques et entrées-sorties

Le système d'exploitation prend en charge (unifie et contrôle) l'accès aux différentes ressources matérielles (périphériques) par le biais des pilotes (*drivers*)

Définition : un pilote est un programme qui effectue les opérations de base pour un type de matériel donné.

Exemples :

- lecture ou écriture d'un secteur sur un disque
- carte son, graphique, réseau
- ...

Fichiers

Système de fichiers :

- organisation arborescente (répertoires, sous-répertoires, ...)
- stockage sur un support physique (partition d'un disque, clé USB, CD, ...), ou accessibles par le réseau
- droits d'accès assurant la protection des données
- mécanisme de résistance aux pannes (journalisation, systèmes RAID, etc.)

Le système d'exploitation gère la lecture et l'écriture dans le système de fichiers en tenant compte des droits d'accès aux fichiers des utilisateurs et applications.

Plan

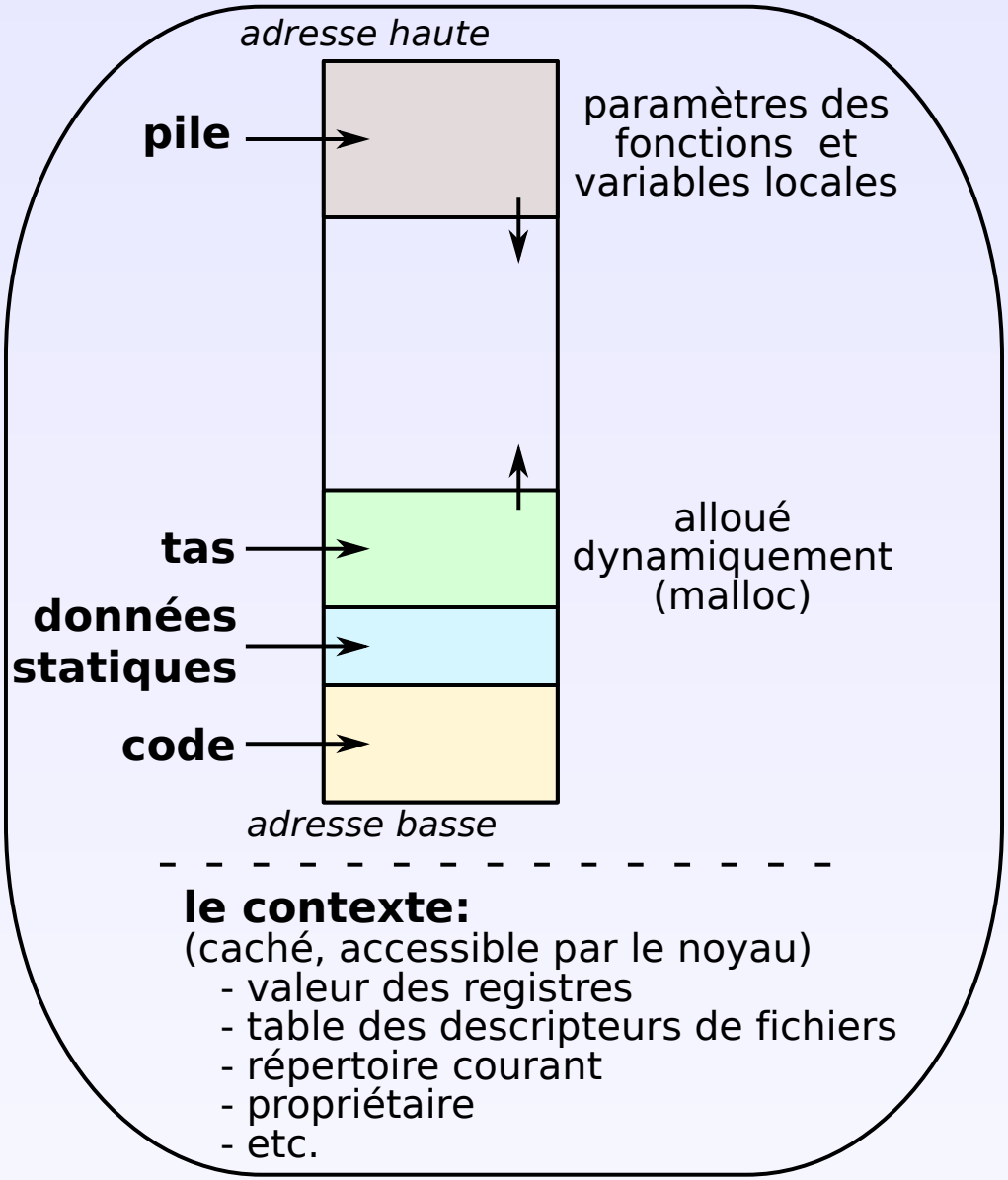
Généralités sur les processus

Définition : un processus est une entité abstraite qui représente un programme en cours d'exécution.

Un processus est défini par :

- un ensemble d'instructions (code du programme)
- un espace mémoire pour les données de travail (pile, tas)
- divers ressources
  - descripteurs de fichiers ouverts, ports réseau, etc.
- des droits d'accès

Espace mémoire d'un processus UNIX



Modes de fonctionnement

Il existe deux modes de fonctionnement :

- le mode noyau (privilégié)
  - toutes les opérations sont autorisées
- le mode utilisateur (protégé)
  - l'accès à la mémoire physique et aux périphériques est protégé

Basculement du mode utilisateur vers le mode noyau :

- sollicitation du système pour une opération particulière, appel système

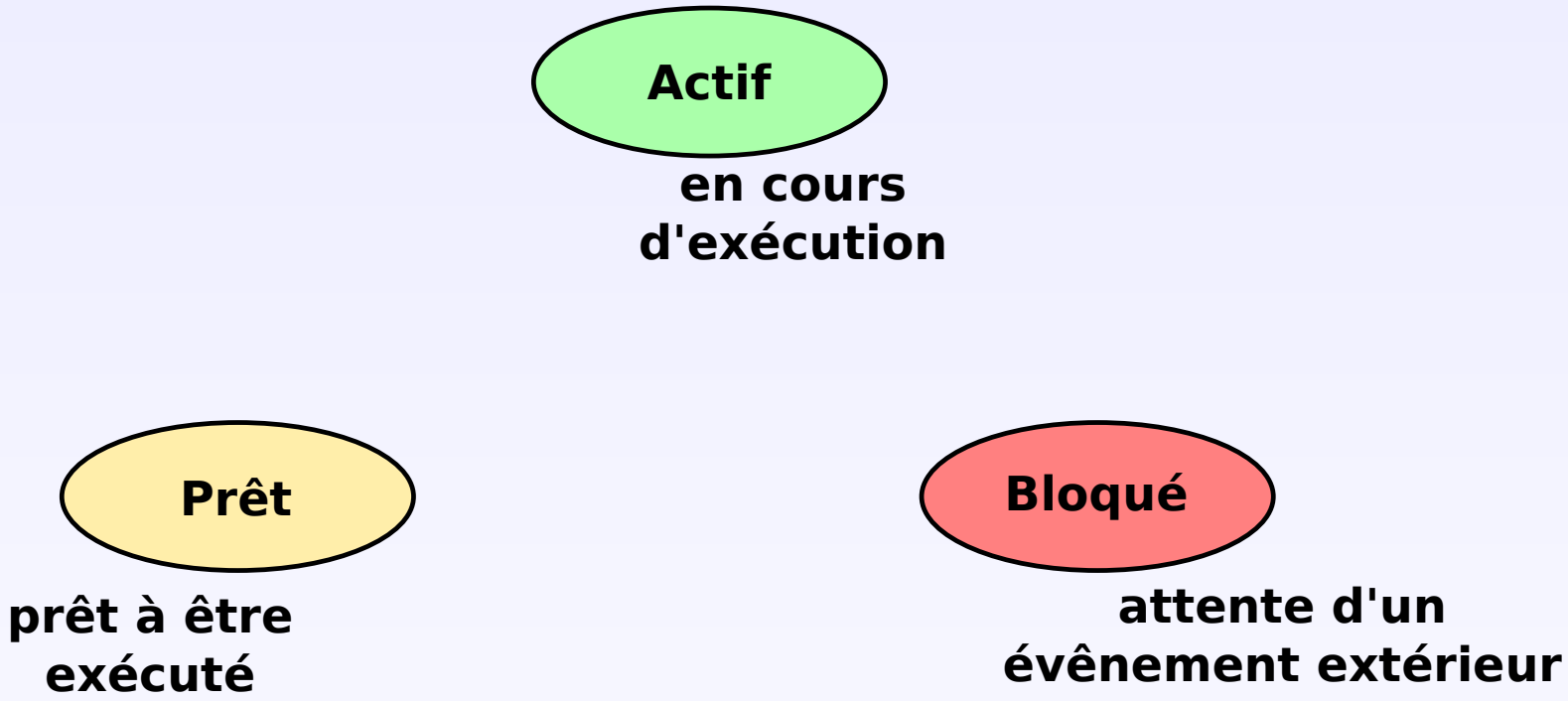
  ex : écriture dans un fichier, allocation mémoire, etc.
- interruption du système

  ex : suite à une erreur mémoire, un signal d'horloge, etc.

États d'un processus

3 états possibles :

- actif : en cours d'exécution
- bloqué : en attente d'un évènement
- prêt : ni actif, ni bloqué



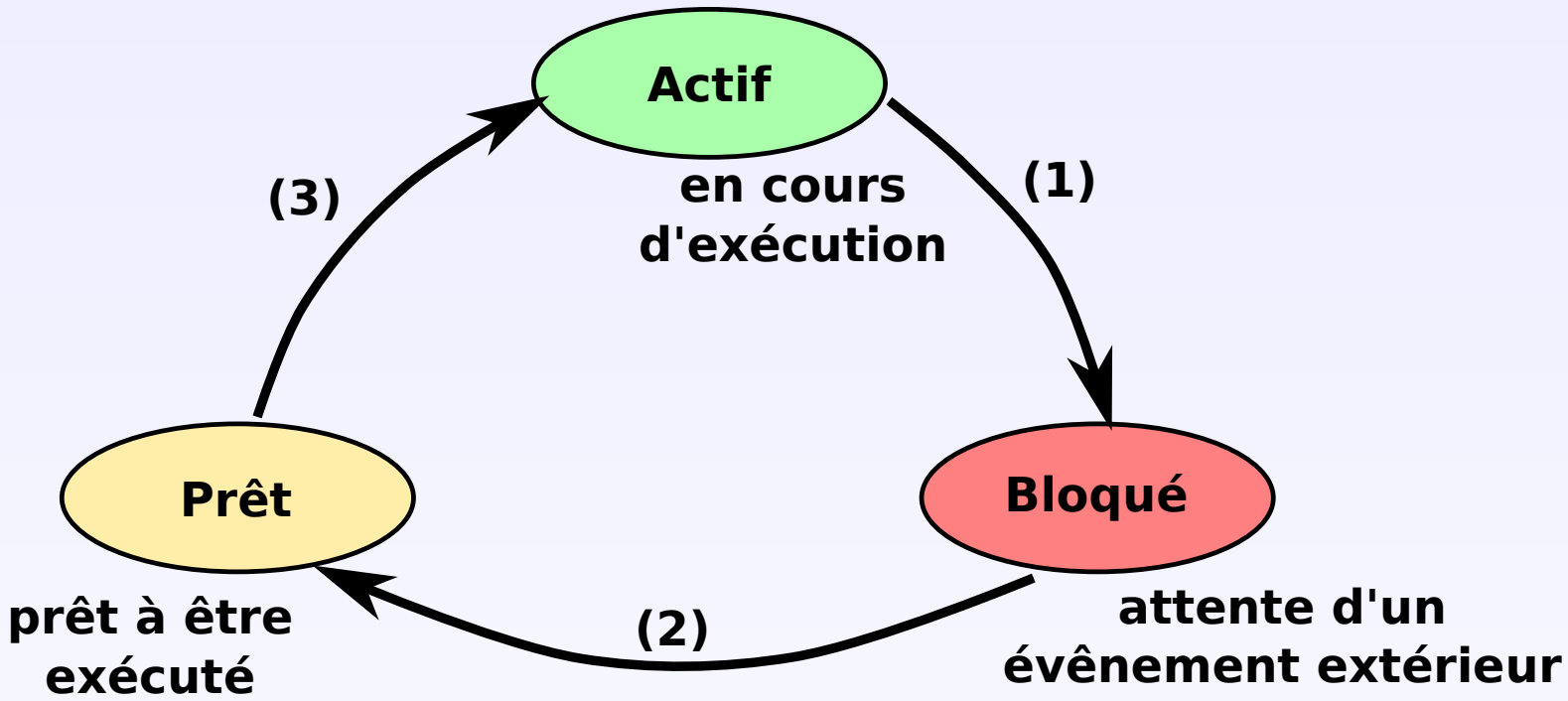
Changements d'états

Un processus passe de l'état :

(1) actif à bloqué lorsqu'il demande une opération d'E/S.

(2) bloqué à prêt quand l'opération d'E/S est terminée

(3) prêt à actif lorsqu'il est choisi par le système



Changements d'états

Dans un système multitâche, plusieurs processus peuvent être prêts

- l'ordonnanceur (*scheduler*) est en charge de désigner le processus à exécuter.
- différentes politiques d'ordonnancement sont possibles.

Définition : l'ordonnanceur est le composant du système qui défini le prochain processus à activer.

Système multitâche préemptif / coopératif

Pour obtenir un partage de temps processeur plus équitable, un processus actif doit pouvoir *céder sa place*.

- exp : monopolisation du processeur par un processus qui ne fait jamais d'E/S.

Pour cela, le processus actif peut

- rendre volontairement la main

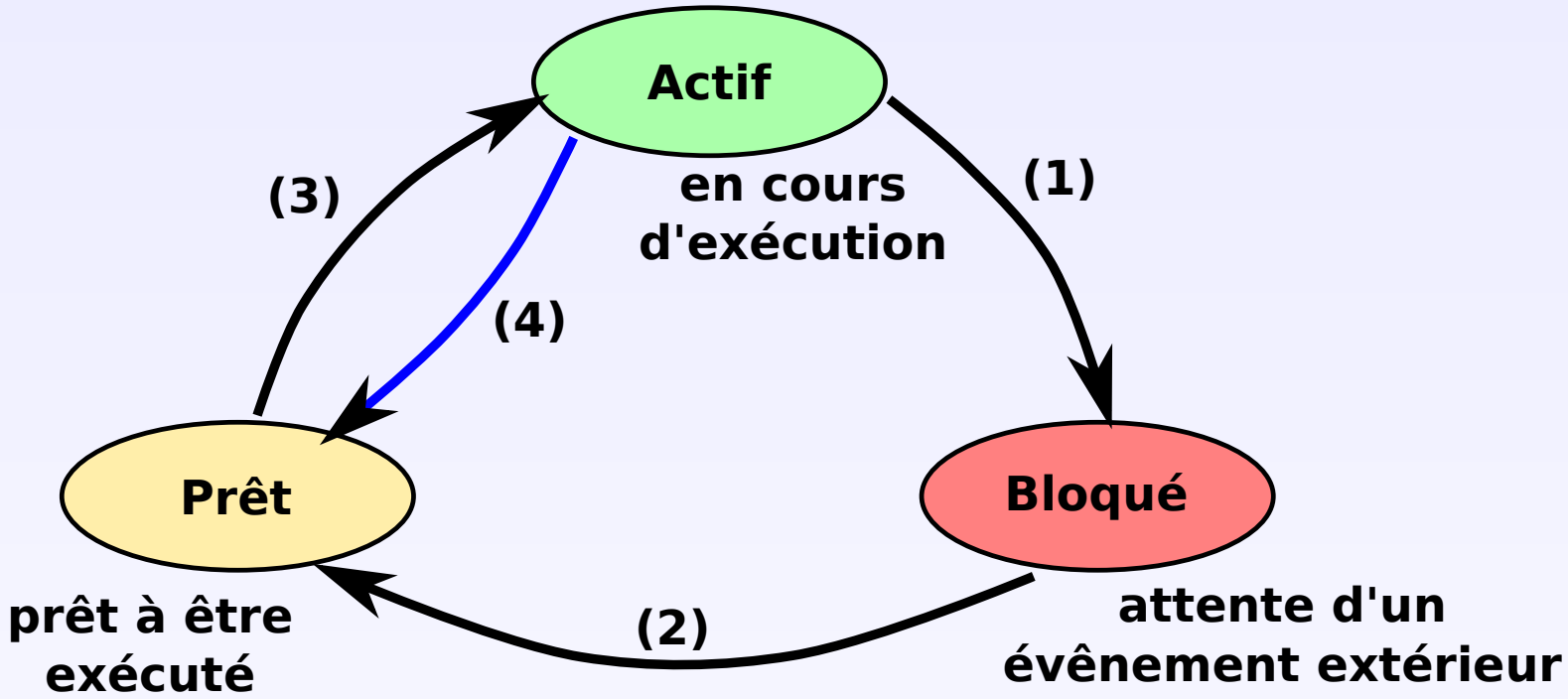
➔ système coopératif, sans réquisition

- être interrompu au bout d'un quantum de temps

➔ système préemptif, avec réquisition

Système multitâche préemptif / coopératif

Pour obtenir un partage de temps processeur plus équitable, un processus actif doit pouvoir *céder sa place*.



Commutation du contexte

Le contexte d'un processus contient toutes les informations relatives à l'exécution du processus (valeur du compteur ordinal, des registres d'états, du pointeur de pile, allocations mémoire, etc.)

Lors des changements d'état, le contexte du processus :

- qui était actif est sauvegardé
- qui devient actif est restauré

➔ reprise de l'exécution où elle a été interrompue.

- sauvegarde du contexte : structure de données appelée PCB (Process Control Block), lors du basculement d'état.
- stockage des PCB des processus présents sur le système dans la table des processus.

Plan

Notions d'interruption et système réactif

Le fonctionnement des systèmes d'exploitation contemporains s'appuie sur la notion d'interruption.

Interruption : signal matériel qui modifie la séquence normale d'exécution des instructions.

➔ système réactif, répond aux évènements causés par l'environnement

Une interruption déclenche

- le passage en mode noyau
- la sauvegarde de l'état du programme (quelques registres)
- l'exécution de la routine de traitement de l'interruption.

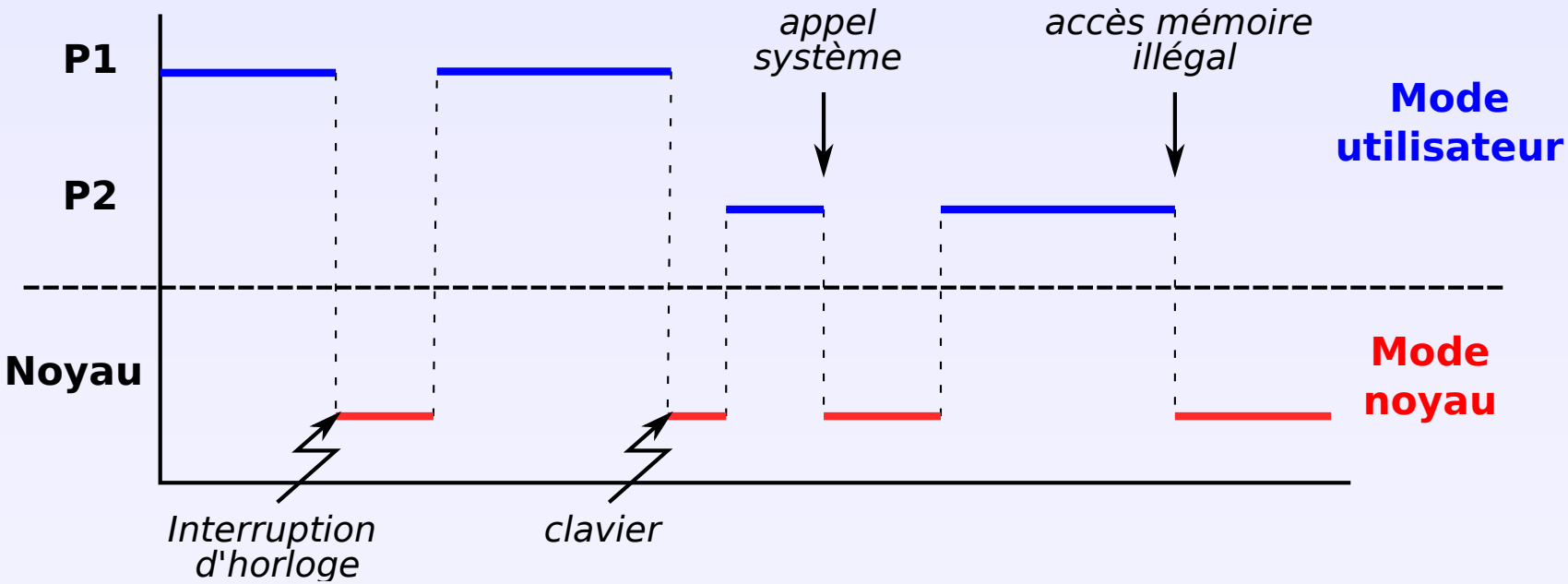
Dans un système multitâche

Les interruptions sont causées par :

- les périphériques (exp : fin d'exécution de requête)
- des signaux d'horloge
  - réguliers, utilisés pour l'ordonnancement
- des évènements extérieurs
- des déroutements en cas d'erreur (accès illégal à la mémoire, division par zéro, etc)
- des interruptions logicielles provoquées par instruction spéciale
  - sollicitation du noyau pour un service particulier

  ➔ appel système

Interruptions et système multitâche



Exemples

Déroulement d'une interruption disque

1. passer le processus actif à l'état prêt
2. déterminer la cause de l'interruption (ex : fin de lecture)
3. trouver le processus demandeur (bloqué)
4. lui transférer les données reçues
5. le mettre à l'état prêt
6. activer un des processus prêts

Interruption d'horloge

- quantum de temps épuisé ➔ ordonnancement avec réquisition (*preemptive scheduling*)

1. remettre le processus actif à l'état prêt
2. activer un des processus prêts

Plan

Ordonnancement

L'ordonnanceur à la charge de désigner le processus à exécuter (ordonnancer).

- il consulte la liste des processus prêts et en choisit un pour l'activation

→ applique une politique d'ordonnancement

Comment définir/choisir une bonne politique d'ordonnancement ?

Nombreux critères :

- équité
- efficacité
- minimiser le temps de réponse
- minimiser le temps d'exécution
- maximiser le rendement,
- maximiser l'occupation du/des processeur(s),
- ...

➔ objectifs parfois contradictoires
➔ comportement des processus ne peut être prévu
➔ importance des critères fonction du contexte

➔ pas de politique optimale, utilisation d'heuristiques

Ordonnancement circulaire

Autres dénominations : tourniquet, round-robin, FIFO (*Fisrt In First Out*), ...

Principe :

- liste circulaire des processus prêts
- à la fin de son quantum de temps, un processus actif est placé en fin de liste, le premier élément de la liste est ordonnancé.

Propriétés :

- équité garantie
- choix du quantum : compromis
  - trop court, nombreux changement de contexte, perte de temps
  - trop long, dégradation du temps de réponse

Ordonnancement avec priorités

On affecte une priorité (numérique) à chaque processus.

Principe :

- choix du processus le plus prioritaire
- dans la même classe de priorité : FIFO

Propriétés

- risque de famine pour les classes de priorité inférieure

➔ remédier aux problèmes d'équité en faisant évoluer les priorités avec le temps (exp : baisse en fin de quantum).

Files multiples

On définit des classes de processus

Principe :

- à chaque classe correspond une liste (circulaire) de processus
- chaque classe est sélectionnée régulièrement

Propriétés

- respect des priorités
- évite les famines

D'autres alogorithmes :

- Plus Court Temps d'Exécution
- Plus Court Temps Restant