Programmation multi-tâche et programmation distribuée

Hinde Bouziane (bouziane@lirmm.fr)

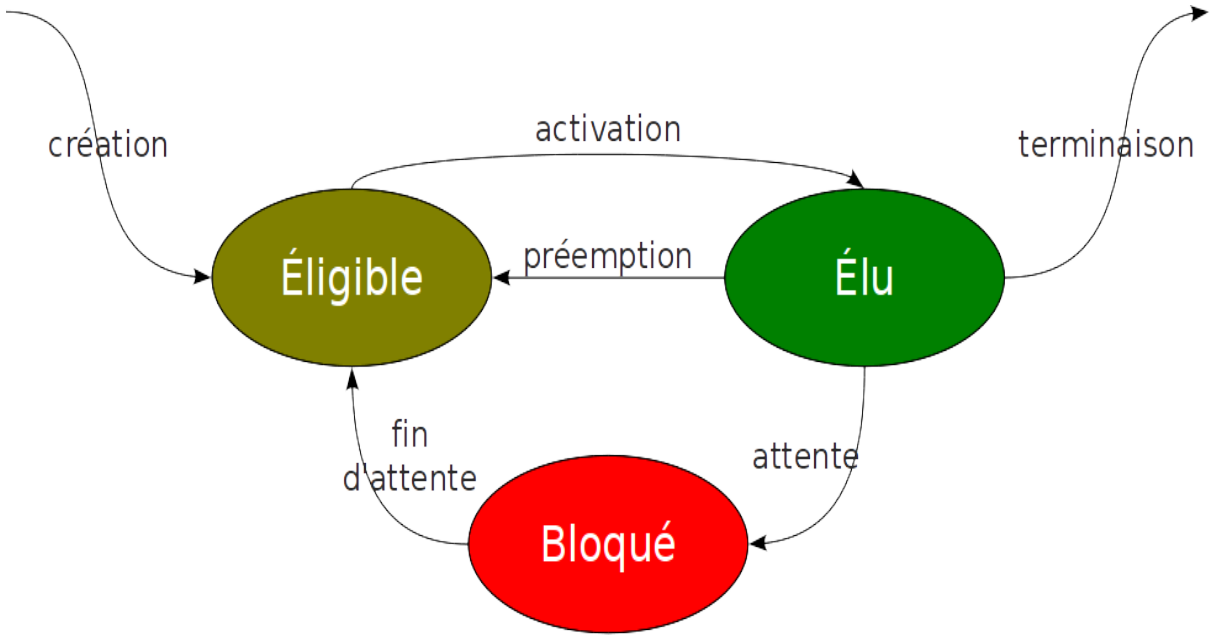
Introduction

Qu'est-ce que le multitâche ?

- Commençons par un système d'exploitation multitâche.
- Un OS est multitâche s'il permet d'exécuter, de façon apparemment simultanée, plusieurs processus.
- Processus : programme en cours d'exécution.
- Simultanéité apparente : le résultat de l'alternance rapide d'exécution des processus présents en mémoire (temps partagé).
- Le passage de l'exécution d'un processus à un autre est appelé commutation de contexte.
- Une commutation peut être initiée par un programme (multitâche coopératif) ou par l'OS (multitâche préemptif). Dans cette UE : multitâche préemptif (on laisse l'ordonnanceur de l'OS gérer).
- Le multitâche s'applique bien sur un système monoprocesseur.

Comment ça marche ?

Cycle de vie d'un processus



Programmation multitâche

- Appelée aussi programmation concurrente.
- Concurrence : potentiel pour l'exécution simultanée (en parallèle) de plusieurs codes sur une même machine.
- Nous parlons de programmation multitâche pour mettre en oeuvre une application composée d'un ou de plusieurs programmes.
- Possibilités :
  - Programmation multi-processus : aboutissant à une exécution simultanée de plusieurs processus (dits processus lourds) issus d'un même programme ou de programmes différents.
  - Programmation multi-thread : aboutissant à une exécution simultanée de plusieurs fils d'exécution (dits threads ou processus légers) au sein d'un même processus.
  - programmation à la fois multi-processus et multi-thread.

A quoi ça sert ?

- Initialement, à optimiser l'utilisation du matériel, en offrant plusieurs fonctionnalités sur un ordinateur ou en parallélisant les I/O avec le CPU
- Aujourd'hui, très largement utilisée et essentielle :
  - Accès simultanés à un serveur Web
  - Utilisation d'un même poste par plusieurs utilisateurs
  - Accès simultanés à un serveur Web
  - Traitements complexes d'images (pour accélérer les calculs)
  - Transferts simultanés de fichiers (exemples : réseaux P2P).
  - Jeux vidéos et réalité virtuelle
  - Simulations scientifiques (prévisions de la météo, biologie, ...)
  - Robotique (temps réel)
  - etc.

Et la place de la programmation distribuée ?

- Programmation réseau pour la mise en oeuvre d'applications de type client-serveur.
- Un exemple attractif impliquant très souvent la programmation multitâche (coté client et/ou serveur).
- Peut être intégrée dans la programmation multi-processus, mais non plus pour une exécution sur une seule machine mais sur plusieurs machines reliées par un réseau (Remarque : un processus client et un processus serveur s'exécutent en parallèle).

Contenu du cours

- Programmation distribuée : communications distantes UDP/IP et TCP/IP
- Programmation multi-thread.
- Programmation multi-processus (IPC).

Remarques

- Application des concepts et outils étudiés en langage C
  - ne signifie pas que ces concepts n'existent pas dans d'autres langages
  - faire la différence entre les propriétés d'un concept et leur implémentation
- Il est primordial de passer par une étape de réflexion et raisonnement avant l'étape d'implémentation
  - définir des algorithmes avant de coder
- Les illustrations dans le cours ne sont que des exemples
  - savoir s'adapter à d'autres contextes

Bibliographie

J.M. Rifflet, J.B. Yunès *Unix, Programmation et communication*, Dunod, 2003

Joëlle Delacroix *Linux, Programmation système et réseau*, 4^{th} edition, Dunod, 2012

Andrew Tanenbaum, *Réseaux*, $5^{ème}$ édition, Pearson Éducation, 2011
Computer Networks, 5^{th} edition.