

Projet de Conception de Systèmes d'Exploitation

<http://chamilo2.grenet.fr/inp/courses/ENSIMAG4MMPCSEF/document/projet.html>

ISI-1 : Christophe RIPPERT

ISI-2 : Sébastien VIARDOT

ISSC : François BROQUEDIS

SLE : Matthieu BRIEDA

Prenom.Nom@Grenoble-INP.fr

Bilan séance 2

- **Implantation de la notion de temps**
 - **Utilisation d'un matériel dédié :**
 - **Horloge, contrôleur d'IT**
 - **Gestion d'une source d'interruption**
 - **Initialisation d'une table système**
 - **Programmation d'un traitant d'IT**
- **Dans la séance 3 et les suivantes :**
 - **Utilisation du même mécanisme pour implanter la notion de temps partagé (*time sharing*) entre des processus**

Temps partagé

- **On travaille sur un processeur (virtuel) mono-cœur :**
 - **Pas de multi-cœur dans ce cours**
 - **Chaque processus pense avoir son processeur (multi-tâches transparent)**
- ⇒ **On partage le processeur en entrelaçant suffisamment vite l'exécution des processus pour donner l'impression à l'utilisateur qu'il y a un processeur / processus**

Rappel : processus

- **Processus = une instance en mémoire d'un programme**
- **Ici : mémoire commune (proc. légers)**
- **Un processus comprend (en résumé) :**
 - **Du code (zone `.text` du programme)**
 - **Des données statiques (zone `.data`)**
 - **Une zone de mémoire dynamique (*heap*)**
 - **Une pile d'exécution (*stack*)**
 - **Des méta-données (nom, état, *etc.*)**
 - **Un contexte d'exécution = l'état de la machine pendant l'exécution du processus (contenu des registres, structures globales, *etc.*)**

Changement de contexte

- **Entrelacer l'exécution des processus :**
 1. Proc1 s'exécute dans son contexte
 2. IT horloge => le noyau prend la main
 3. L'ordonnanceur (*scheduler*) choisi le prochain processus à exécuter
 4. Le noyau sauvegarde le contexte de Proc1 et restaure celui de Proc2
 5. Le noyau passe la main à Proc2 qui reprend exactement là où il s'était arrêté
- **Code fourni (`ctx_sw.S`) à comprendre**

Cette séance

- **On commence avec 2 processus :**
 - Créés statiquement au démarrage du noyau
 - Qui ne se terminent jamais
- **Le changement de processus sera explicite :**
 - Appel à `ctx_sw` dans le code du processus
 - Ordonnancement « collaboratif »
 - Pas d'utilisation de l'horloge dans cette séance
- **Politique d'ordonnancement simple**
 - Algorithme du tourniquet (*round-robin*)
 - C'est à dire « chacun son tour »
- **Deux états pour les processus :**
 - « Elu » : celui qui s'exécute
 - « Activable » : celui qui attend son tour

Séances 4, 5 et 6

- **Généralisation à N processus**
 - **Changement de processus basé sur l'horloge**
 - **Ordonnancement « préemptif »**
 - **Endormissement des processus**
 - **Terminaison des processus**
 - **Création dynamique de processus**
- **Deux versions du sujet (et de l'examen) :**
- **ISI : système « dynamique »**
 - **ISSC & SLE : système « embarqué »**
- **Exactement les mêmes notions !**