GRENOBLE INP UGA Ensimag

École nationale supérieure d'informatique et de mathématiques appliquées

Projet de conception de système d'exploitation

Ensimag 2A filière ISI

Christophe.Rippert@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 cette séance et les suivantes :**
  - **Utilisation de ce mécanisme pour implanter du « temps partagé » (*time-sharing*) entre des processus**

Temps partagé

- On travaille sur un processeur (virtuel) mono-coeur :
  - Pas de multi-coeur dans ce cours
- Chaque processus pense avoir son processeur dédié
  - Multi-tâche transparent pour les processus

=> On entrelace l'exécution des processus suffisamment vite pour donner l'illusion à l'utilisateur que les processus s'exécutent en parallèle

Rappel : processus (1/2)

- Un processus == une instance en mémoire d'un programme en cours d'exécution
- Ici : mémoire commune => processus légers (*threads*)
- Un processus comprend :
  - Une partie « visible » par le programmeur (assembleur) :
    - Du code (zone `.text`) du programme
    - Des données statiques (zone `.data`)
    - Une zone de mémoire dynamique (`malloc` / `free`) : le tas (`heap`)
    - Une pile d'exécution (`stack`)

Rappel : processus (2/2)

- Et une partie « invisible » gérée par le système :
  - Des méta-données :
    - Nom du processus
    - Son état (élu, activable, endormi, *etc.*)
    - *etc.*
  - Un contexte d'exécution c.a.d 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… IT horloge => le noyau prend la main
  2. L'ordonnanceur (*scheduler*) choisit le prochain processus
  3. Le noyau sauvegarde le contexte de Proc1 et restaure celui de Proc2 (sauvegardé lorsque Proc2 avait perdu la main)
  4. Le noyau passe la main à Proc2 qui reprend son exécution là où il s'était arrêté
- **Code fourni (`ctx_sw.S`) dont il faut comprendre le principe**

Implantation pas à pas (1/2)

- On commence par deux processus :
  - Créés statiquement au démarrage du système
  - Qui ne se terminent jamais
- Le changement de processus sera explicite :
  - Appel à `ctx_sw()` dans le code (ordonnancement collaboratif)
  - Pas d'utilisation de l'horloge initialement
- Politique d'ordonnancement simple :
  - Algorithme du tourniquet (*round-robin*)
  - C'est à dire « chacun son tour »

Implantation pas à pas (2/2)

- **Deux états possibles pour chaque processus :**
  - **« Élu » : le processus en cours d'exécution**
  - **« Activable » : le processus qui attend son tour**
- **Puis on enrichit le système :**
  - **Généralisation à N processus**
  - **Changement de processus basé sur l'horloge (ordonnancement préemptif)**
  - **Endormissement des processus**
  - **Terminaison et création dynamique des processus**