Gestion des processus et changement de contexte

Plan

- Gestion des processus
- Changement de contexte
- Création d'un processus
- Ordonnancement

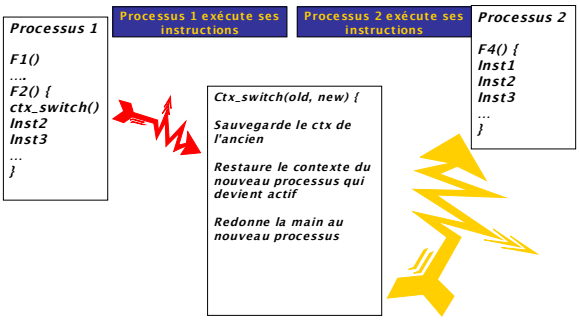
Processus

- Nombre limité de processus (256)
  - souvent : table des processus
- Caractérisés par un ensemble d'informations
  - pid (unique)
  - actif, activable,bloqué, …
  - priorité
  - programme de démarrage (fonction initiale)
- Ordonnancement (*Scheduling*)
  - en fonction de la priorité
  - *time-sharing* de SCHEDFREQ Hz (constante du système)
- Changement de contexte
  - technique pour permettre le temps partagé
- Naissance et mort

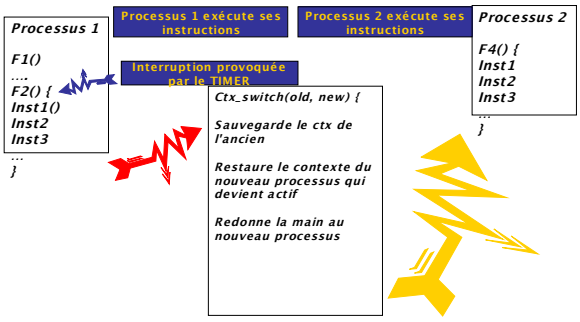
Structure de données du processus

- Informations
  - Identifiant : pid
  - Priorité
  - Contexte (registres, piles, ...) caractérisant l'exécution
  - Informations diverses (nom du processus, pid du père, liste des fils , lien, …)
- État
  - Caractérise l'avancement du processus et ce que le système peut faire avec ce processus
    - Actif (élu) : c'est le processus en cours d'exécution
    - Activable (éligible) : le processus est dans une liste de processus en attente d'être activé
    - Bloqué : le processus est en attente d'être débloqué par un outil de synchronisation (sémaphore)
    - Endormi : le processus doit être réveillé par *sleep*
  - Utilisé principalement par l'ordonnanceur pour savoir quel est le prochain processus pouvant être exécuté

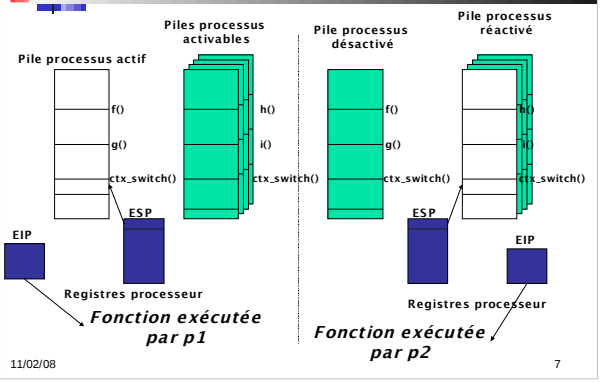
Changement de processus actif



Changement de processus actif



Le changement de contexte



Contexte du processus

- Valeur des registres du processeur
  - registres banalisés : `eax`, `ebx`, `ecx`, `edx`, `esi`, `edi`
  - registres de pile : `esp`, `ebp`
  - compteur d'instructions : `eip`
  - Flags
  - Voir dans la doc Intel lesquels doivent être sauvés
- Type `contexte_execution`
  - structure ou tableau contenant l'ensemble des valeurs de registre

Réalisation du changement (i)

- Changer de contexte = changer de processus en cours d'exécution

1. Sauvegarder le contexte du processus en cours
   - stocker les registres dans la structure de données `contexte_execution`
2. Charger le contexte du processus qui va s'exécuter
   - restaurer les registres à partir de la structure de données `contexte_execution` ➡ change le sommet de pile

Réalisation du changement (ii)

- Appel du changement de contexte

`changement_ctx(contexte *a_sauver, contexte *a_restaurer)`

  - `a_sauver`
    - va contenir la valeur des registres durant l'appel
    - champ contexte du processus qui va passer de actif à activable
  - `a_restaurer`
    - contient la valeur qu'auront les registres du processeur à la fin de l'appel
    - champ contexte du processus qui va passer de activable à actif

Remarques et aide

- Les valeurs des paramètres sont stockées sur la pile
  - attention à ne pas changer de pile trop tôt !
    - changer de pile = changer la valeur de `esp`
    - **`Ou garder l'ancienne valeur de esp dans un registre`**
- Le changement de contexte se passe **toujours** dans la fonction `changement_ctx`
  - quand on restaure le contexte on sait à quel endroit du code on était au moment où on a pris la valeur des registres
  - pour sortir de la fonction `changement_ctx` on peut faire un simple `ret`
    - il faut que le pointeur de pile soit correct !

Création de processus

- Étapes de création
  - choisir son pid
    - càd trouver une entrée libre dans la table
  - affecter ses informations d'état
    - priorité, père, …
    - état initial : activable
  - créer son contexte d'exécution
    - allouer une pile (éventuellement la remplir)
    - affecter une valeur initiale aux registres du contexte d'exécution (important : la pile !)

Création du contexte initial

- Un processus créé doit exécuter une fonction avec un paramètre
- Le processus sera démarré par un appel à **`changement_ctx`**
  - pour que la sortie de l'appel de **`changement_ctx`** se fasse bien ➡ pile correcte
- Nécessité de créer une pile avec le paramètre et l'adresse de la fonction à exécuter installés

Stratégie d'ordonnancement

- Décide quel processus sera exécuté après épuisement du quantum ou changement d'état d'un processus
- Politique FIFO
  - les processus les plus prioritaires s'exécutent **toujours** avant les moins prioritaires
  - stratégie non équitable : famine possible
  - partage de temps **équitable** entre processus de **même priorité**
  - s'applique aussi aux opérations de synchronisation (le plus prioritaire « sort » du sémaphore le premier)
- Implantation
  - liste des processus activables triée par priorité
  - utilisation du paquetage de gestion de file donné

Fonctionnement du temps partagé