Architecture des Ordinateurs et Systèmes d'Exploitation

Cours n°10

Les Processus : Introduction / Ordonnancement Les processus sous Unix

INSA ROUEN
Ph. Leray

asi Architecture des Systèmes d'Information
3ème année

Qu'est-ce qu'un processus?

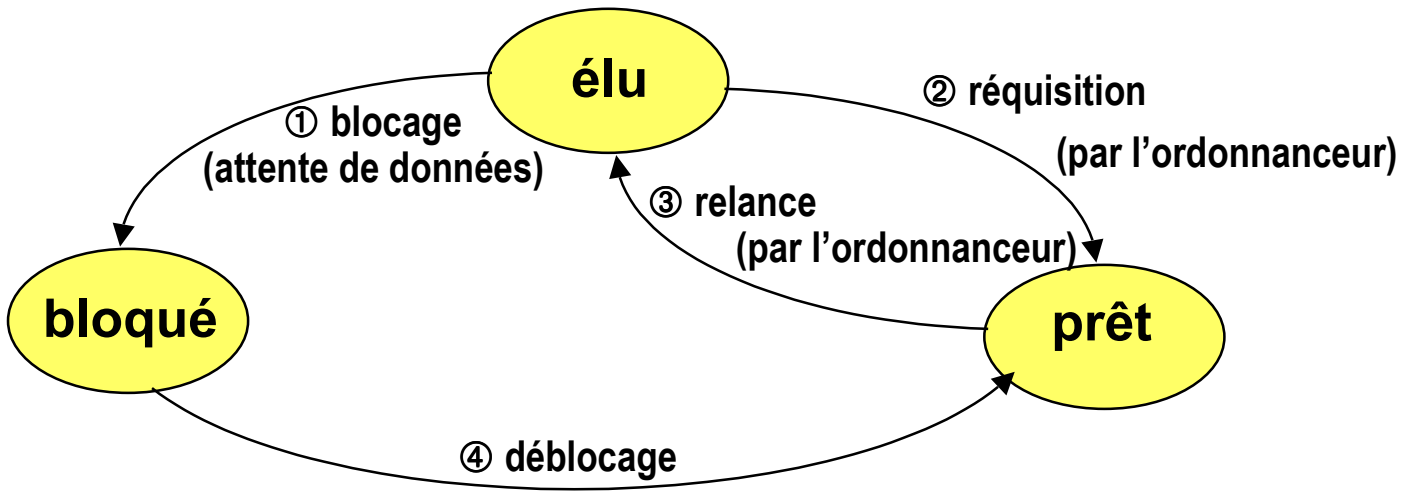
- **Processus = unité d'exécution (unité de partage du temps processeur et de la mémoire)**

- **Processus ≠ Programme :**
  - **un programme peut être exécuté plusieurs fois et se trouver dans plusieurs unités d'exécution en même temps**
  - **le processus doit connaître à chaque instant**
    - » **le code du programme**
    - » **le pointeur d'instruction**
    - » **l'état de la pile**
    - » **les variables**
  - **on peut conserver un processus en changeant le code qu'il exécute**

- **Le SE doit ordonnancer les processus (*scheduler*)**

Les différents états d'un processus

- **Un processus peut être dans 3 états possibles :**
  - **élu (en cours d'exécution)**
    processus OK, processeur OK
  - **prêt (suspendu provisoirement pour qu'un autre processus s'exécute)**
    processus OK, processeur occupé
  - **bloqué (attendant un événement extérieur pour continuer)**
    processus non OK, même si processeur OK



Ordonnancement des processus

- **Plusieurs processus sont prêts à être exécutés**
- **Le SE doit faire un choix … (algorithme d 'ordonnancement) :**
  - **équité : chaque processus doit avoir du temps processeur**
  - **efficacité : le processeur doit être utilisé à 100%**
  - **temps de réponse : l 'utilisateur devant sa machine ne doit pas trop attendre**
  - **temps d'exécution : une séquence d'instructions ne doit pas trop durer**
  - **rendement : il faut faire le plus de choses en une heure**

- **Ordonnancement sans réquisition: un processus est exécuté jusqu'à la fin**
  - **inefficace et dangereux (ex: exécution d'une boucle sans fin…)**

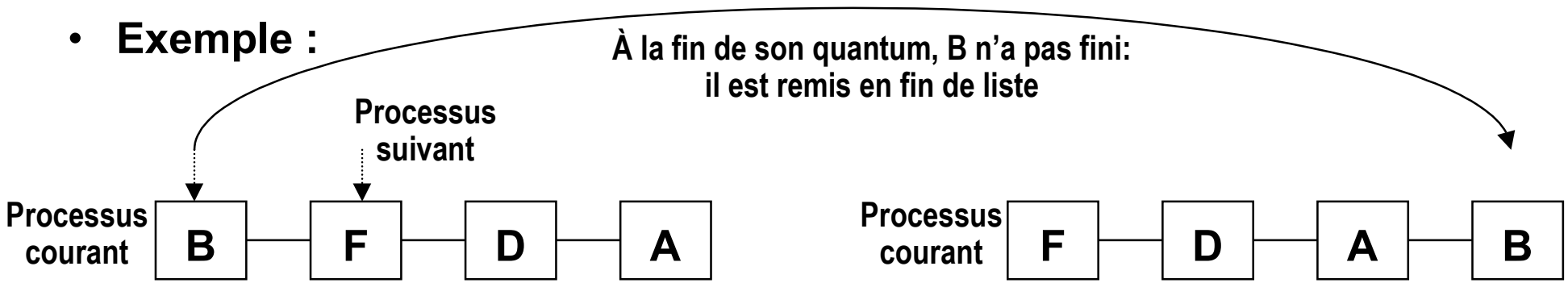
- **Ordonnancement avec réquisition :**
  - **à chaque signal d 'horloge, le SE reprend la main, décide si le processus courant a consommé son quota de temps machine et alloue éventuellement le processeur à un autre processus**
  - **il existe de nombreux algorithmes d'ordonnancement avec réquisition**

Ordonnancement circulaire (tourniquet)

- **Chaque processus possède un quantum d'exécution**
  - **Si le processus a fini dans cet intervalle : au suivant !**
  - **S'il n'a pas fini : le processus passe en fin de liste et au suivant !**

- **Problème = réglage du quantum :**
  - **quantum trop petit / commutation (= temps de passage d'un processus à l'autre) : le processeur passe son temps à commuter**
  - **quantum trop grand : augmentation du temps de réponse d'une commande (même simple)**
  - **réglage correct: Quantum/commutation = 5**

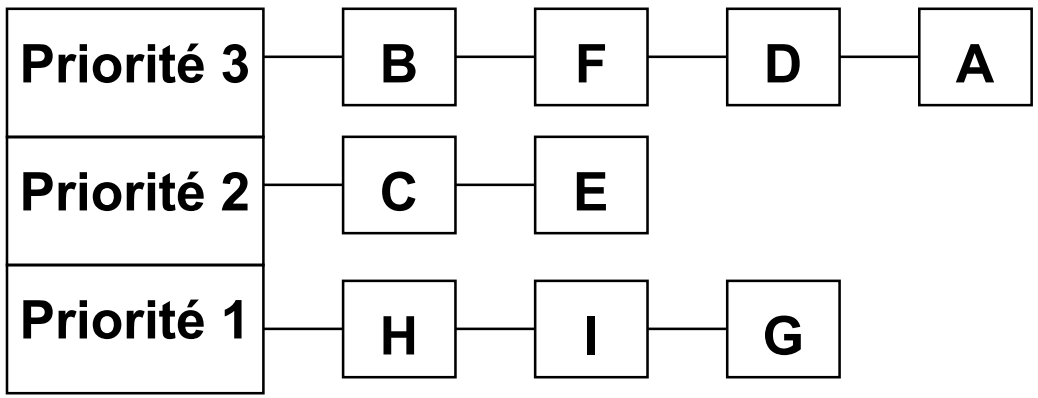
- **Exemple :**



Ordonnancement avec priorité

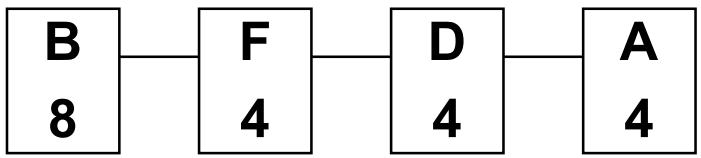
- **Inconvénient du tourniquet = processus de même priorité**

- **Ordonnancement avec priorité :**
  - **plusieurs files d'attente plus ou moins prioritaires**
  - **la priorité d'un processus décroît au cours du temps pour ne pas bloquer les autres files d'attente**

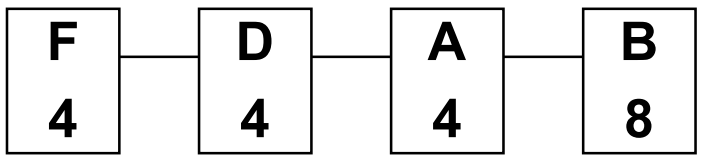


Ordonnancement «plus court d'abord»

- **Les algorithmes précédents marchent bien pour les processus interactifs.**
- **Ordonnancement «plus court d'abord» :**
  - **estimation de la durée de chaque processus en attente**
  - **exécuter le processus le plus court**



• sans ordonnancement :
Tmoyen = 1/4 (Tb+Tf+Td+Ta) = 14
avec Tb=8, Tf=8+4,
Td=8+4+4, Ta=8+4+4+4



• avec ordonnancement :
Tmoyen = 1/4 (Tf+Td+Ta+Tb) = 11
avec Tf=4, Td=4+4,
Ta=4+4+4, Ta=8+4+4+4

- **Méthode optimale si les processus sont disponibles immédiatement**

Ordonnancement garanti

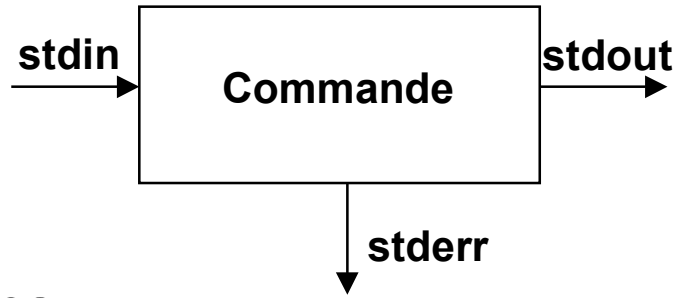
- **Approche différente des précédentes**
- **Garantie de service à l'utilisateur**
- **Principe :**

  si n utilisateurs connectés, chacun reçoit 1/n du temps processeur (Tréservé)

- **Pratique :**
  - **pour chaque processus, calcul de ratio = Tconsommé/Tréservé**
  - **exécution du processus de ratio le plus faible jusqu'au moment où celui-ci arrive au niveau d'un autre processus**

Unix et les processus : avant-propos

- **Exécution d'une commande :**



- **Exécution de plusieurs commandes :**

  C1 ; C2 ; C3 ; … (commandes indépendantes exécutées à la suite)

- **Redirections :**

  commande < fichier1
  commande > fichier2 (>>)
  commande >& fichier3 (>>&)

- **Connexions :**

  ex: who>temp ; wc -l < temp ; rm temp

  (| = tube = *pipe*) ⇒ who | wc -l *(stdout1 = stdin2)*

- **Ex° conditionnelle :** c1 && c2 (exécute c2 ssi c1 réussit)
  c1 || c2 (exécute c2 ssi c1 échoue)

Caractéristiques des processus

- **Processus fils / père :**
  - **ex : Le shell est un processus comme les autres**

    Chaque commande exécutée correspond à la création d'un processus « fils » par rapport au shell (« père »)
  - **Sous Unix, chaque processus est identifié par :**
    - » **PID (*Processus Identifier*)**
    - » **PPID (*Parent Processus Identifier*)**

- **2 types de processus :**
  - **processus systèmes (*daemons*)**

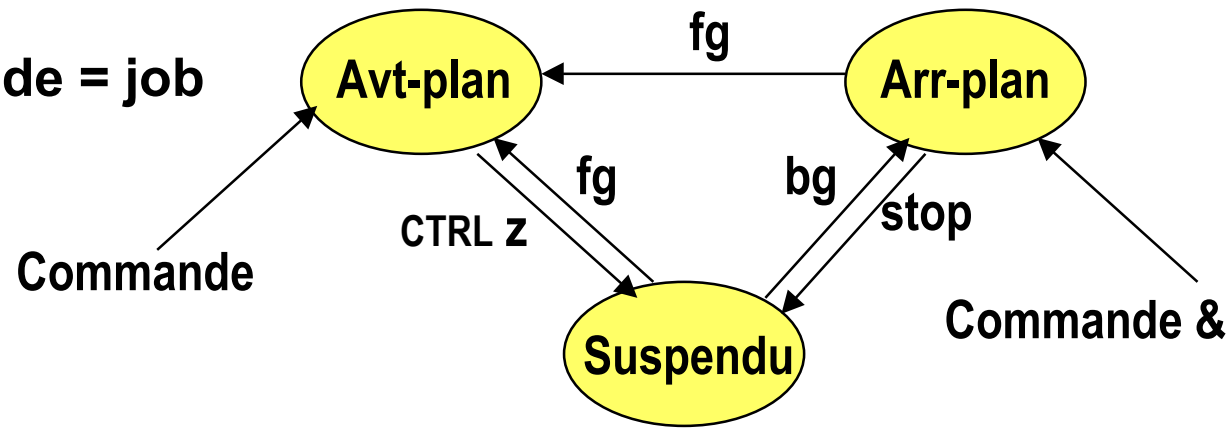
    exécution de tâches générales, souvent contrôlées par root
  - **processus utilisateurs**

Modes d'exécution 1/2

- **Interactif (*foreground*) :** **more /etc/passwd**
  - **le plus fréquent (on tape une commande, on attend un résultat, …)**
  - **interruption de la commande par CTRL C**
  - **suspension de la commande par CTRL Z**

- **Arrière-plan (*background*) :** **grep leray /etc/passwd > f &**
  - **la commande est lancée, mais on rend le contrôle à l'utilisateur**
  - **pas d'interaction avec l'utilisateur**

Exécution d'une commande = job



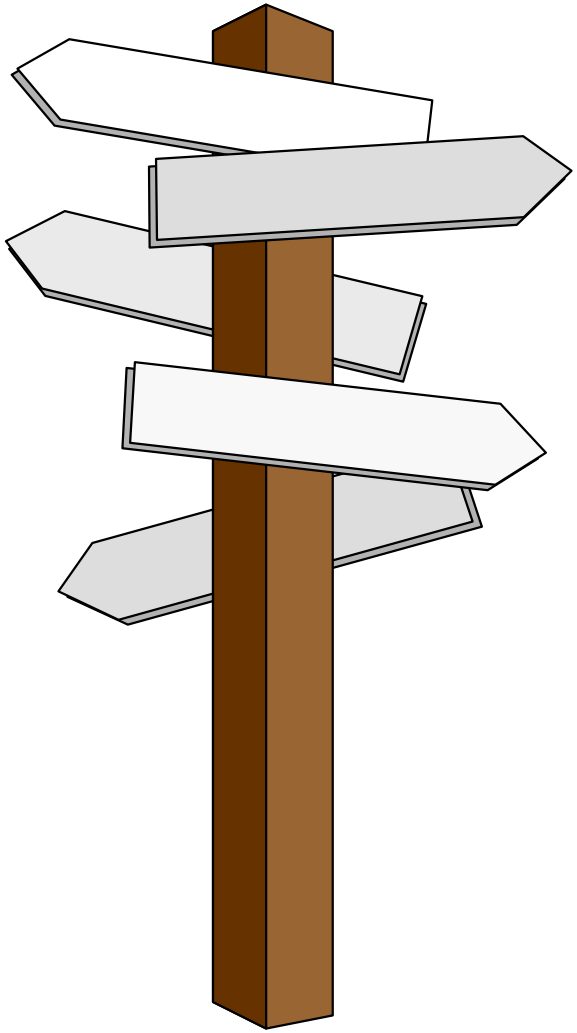
Modes d'exécution 2/2

- **Différé (*at*)**
  - **le fichier de commandes est exécuté à une date fixée**
  - **pas d'interaction avec l'utilisateur**
  - **les résultats peuvent être envoyés à l'utilisateur par e-mail !**

- **File d'attente (*batch*)**
  - **la commande est placée dans une file d'attente**
  - **la file d'attente est vidée en fonction de la charge du processeur**
  - **les résultats peuvent être envoyés à l'utilisateur par e-mail !**

- **Cyclique (*crontab*)**
  - **un fichier spécial contient les tâches à exécuter régulièrement**
  - **un *daemon* scrute sans arrêt ce fichier**

Références



Processus : généralités

- Systèmes d'Exploitation - A. Tanenbaum (InterEditions)

Gestion de processus : UNIX/Linux :

- La programmation Unix - J.M. Rifflet (Ediscience)
- UNIX : Guide de l'étudiant - H. Hahn (Dunod)
- Linux in a nutshell - E. Siever (O 'Reilly)

A suivre :

- TD n°10 : Ordonnancement de Processus
- TP n°10 : Processus sous Unix
- Cours n°11 : Communication inter-Processus
- TP n°11 : Communication inter-Processus (C | Unix)
- Cours n°12 : Interblocage