Cours IFT1215 - Automne 2005
Introduction aux systèmes informatiques

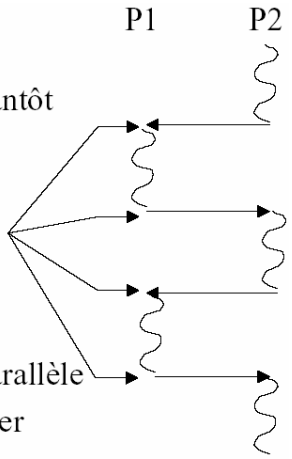
15. La gestion des processus

Professeur:
Victor Ostromoukhov

15 La gestion des processus

Pseudo parallélisme

- Plusieurs programmes en mémoire
- Le processeur exécute les instructions tantôt pour l'un tantôt pour l'autre



- Vu de l'utilisateur => programmes en parallèle
- Gestion complexe, difficile à appréhender

⇨ processus

Notion de processus (1)

- Programme
  - ensemble de modules sources
  - ensemble de modules objets
  - résultat de l'édition de lien
    - ⇨ description des actions à entreprendre
- Processeur
  - entité matérielle capable d'exécuter des instructions
  - parfois aussi entité logiciel (interpréteur,…)
- Processus
  - entité dynamique correspondant à l'exécution d'une suite d'instruction
  - concept abstrait
  - le processeur fait évoluer le processus

Notion de processus (2)

- Système
  - doit distinguer les différents processus
  - doit représenter l'état d'un processus
- État
  - localisation du programme (instructions)
  - données propres
  - variables
  - registres, dont compteur ordinal,...

Exemple (1)

- Informaticien fait un gâteau d'anniversaire pour sa fille
  - programme => recette de cuisine
  - données entrées => ingrédients
  - ressources nécessaires => instruments
  - processeur => informaticien
  - processus => activité de transformation des ingrédients en gâteau
- Fils de l'informaticien se fait piquer par une abeille
  - interruption du travail => sauvegarde de l'état du processus
  - programme => livre de première urgence
  - processus => soin à apporter, calmer son fils
- Reprise du travail de cuisinier
  - restitution de l'état du processus

Exemple (2)

- Fille de l'informaticien annonce de nouveaux invités
  - fabriquer 2 gâteaux
  - 2 processus distincts sur le même programme
  - une seule recette => programme réentrant
  - séparation des données propres de chacun des processus
    - zones de variables distinctes

➪ *processeur virtuel, temps virtuel*

Hiérarchie de processus (1)

- Nombre fixe de processus banalisés
  - 1 par terminal, avec attachement fixe
  - n attribués à la demande pour l'exécution d'une commande
- Création dynamique de processus

```
id := créer_processus (programme, contexte)
```

identité du processus créé — *instructions*, *données* — *état initial du processus*

  - induit une relation créateur-créé => relation père-fils (arbre)
  - fin du père
    - détruire les fils non achevés
    - attendre l'achèvement des fils (problème de contexte)
    - rattachement à un ancêtre (la racine)

Exemple d'Unix

- Création = copie conforme
  - même programme
  - copie des données, des variables, des registres (compteur ordinal)
- Distinction: valeur retournée par l'opération
  - 0 pour le fils
  - identité du fils pour le père

```
id_fils := fork();
si id_fils = 0 alors { instructions du fils }
               sinon { instructions du père }
```

- Fin du père avant fils => fils rattaché à la racine
  - le père peut attendre la fin du fils: `id_fils := wait (&status);`
- Fonction `exec` => changement du programme en cours

  ➪ `créer_processus` = combinaison `fork` et `exec`

Notion de ressource

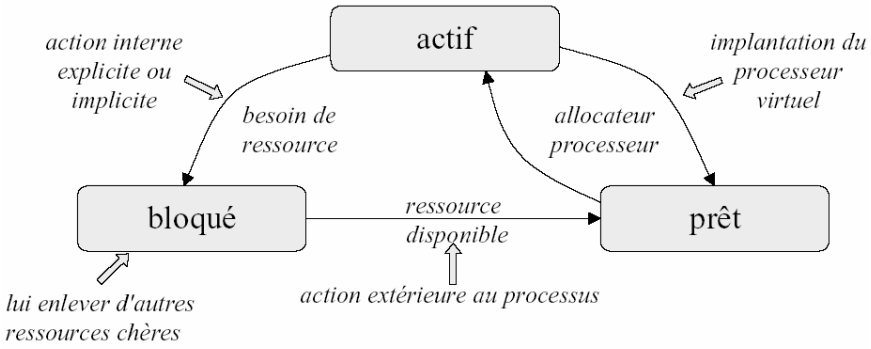
- Sur l'exemple
  - ressources du processus de réalisation 1er gâteau => ustensiles, denrées
  - ressources du processus de réalisation 2ème gâteau => ustensiles, denrées
  - manque de ressources => conflit entre les processus
    - four => assez grand pour 2 gâteaux — 2
    - batteur => un processus à la fois — 1
    - horloge => partagée par tous — ∞

Ressource = toute entité dont a besoin un processus pour s'exécuter

processeur physique, mémoire centrale, périphériques,
données momentanément indisponibles, événement, etc.

- Nombre de points d'accès
  - nombre de processus possédant la ressource à un instant donné
  - si un seul => *ressource critique*, processus en *exclusion mutuelle*

États d'un processus



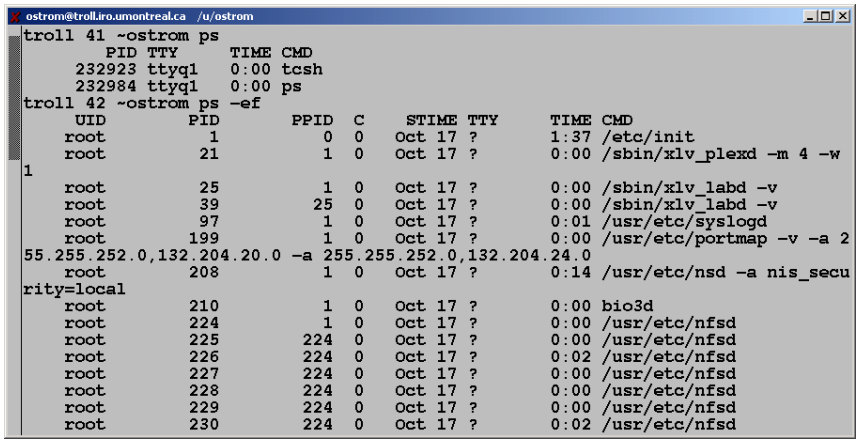
transitions induisent une *déperdition* (*overhead*)

fin du processus => récupérer toutes ses ressources

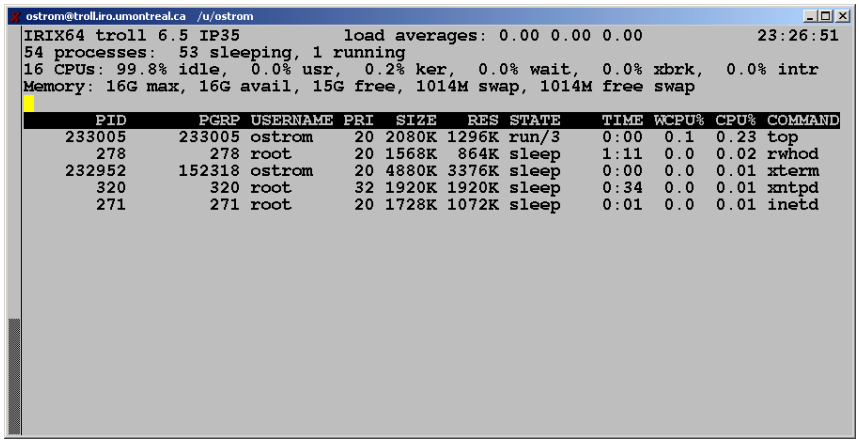
Exemples: ps

```
ostrom@troll.iro.umontreal.ca  /u/ostrom
troll 41 ~ostrom
troll 41 ~ostrom
troll 41 ~ostrom ps
      PID TTY      TIME CMD
   232923 ttyq1    0:00 tcsh
   232984 ttyq1    0:00 ps
troll 42 ~ostrom
```

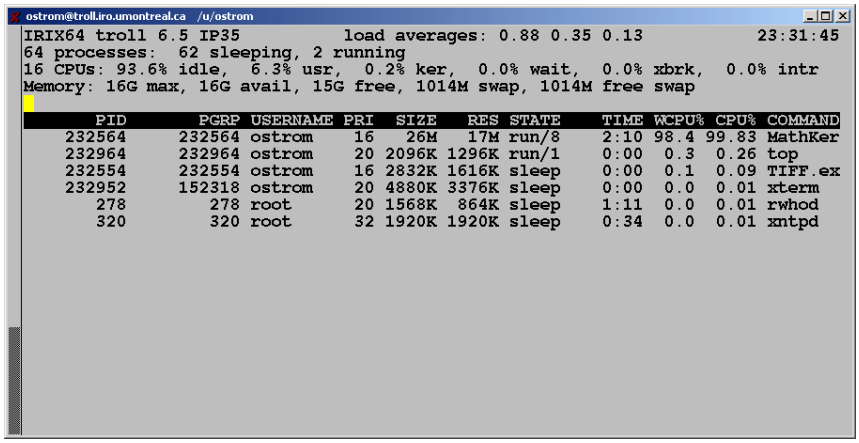
Exemples: ps -ef



Exemples: top

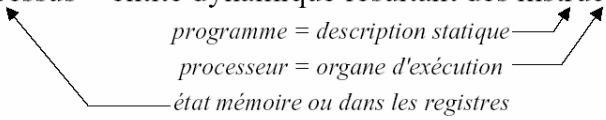


Exemples: top



Conclusions

- Processus = entité dynamique résultant des instructions



- Création dynamique => programme + contexte initial
- Ressource = entité nécessaire à un processus
  - si nombre limité de processus peuvent l'utiliser => contrôle et blocage
- 3 états d'un processus: actif, prêt, bloqué
  - actif → bloqué = action volontaire ou réquisition
  - bloqué → prêt = action extérieure
  - prêt ↔ actif = implantation du processeur virtuel

Résumé

+ Un processus est l'entité dynamique qui correspond à l'exécution d'une suite d'instructions définie par le programme. Le processeur est l'entité qui exécute les instructions pour le compte du processus.

+ Lorsqu'un processus n'a pas le processeur, il ne peut évoluer, et son état doit être conservé en mémoire. Lorsqu'il a le processeur, son état évolue, et est représenté en partie par les registres du processeur.

+ Il peut y avoir un nombre fixe de processus dans un système, ou bien les processus peuvent être créés dynamiquement. Dans ce cas, l'ensemble des processus est structuré en arborescence suivant la relation créateur-créé.

+ La création d'un processus doit définir le programme qu'il doit exécuter, et le contexte initial de cette exécution. Sur Unix, le processus créé est une copie conforme du processus créateur.

+ Une ressource est une entité dont a besoin un processus à un instant donné pour s'exécuter. Elle est caractérisée, en particulier, par le nombre de processus qui peuvent l'utiliser au même moment. Ce nombre peut être illimité, et aucun contrôle n'est nécessaire lors de l'allocation.

+ Si le nombre de processus pouvant utiliser une ressource est borné, il faut contrôler lors de l'allocation que la borne n'est pas atteinte. Lorsque cette borne est égale à 1, on dit que

la ressource est critique, et que les processus sont en exclusion mutuelle pour cette ressource.

+ Les ressources induisent trois états sur les processus: l'état actif lorsque le processus a toutes ses ressources, l'état prêt lorsque le processus a toutes ses ressources sauf le processeur et l'état bloqué lorsqu'il lui manque des ressources autres que le processeur.

+ Le passage de l'état actif à l'état bloqué est en général volontaire, ou à la suite d'une réquisition. Le passage de l'état bloqué à l'état prêt est dû à une action externe au processus.