

Cours IFT1215 - Automne 2005
Introduction aux systèmes informatiques

15. La gestion des processus

Professeur:
Victor Ostromoukhov

15 La gestion des processus

La gestion des processus

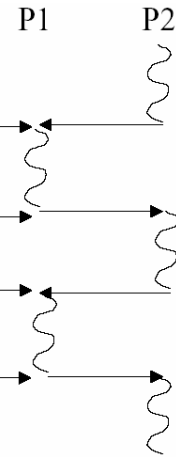
- 15.1. La notion de processus
- 15.2. La hiérarchie de processus
 - 15.2.1. Un nombre fixe de processus banalisés
 - 15.2.2. La création dynamique de processus
 - 15.2.3. L'exemple de Unix
- 15.3. La notion de ressources
- 15.4. Les états d'un processus
- 15.5. Conclusion

Pseudo parallélisme

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

programme de supervision

- 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*

Hierarchie 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
état initial du processus*

↑

données

 - ◆ 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)

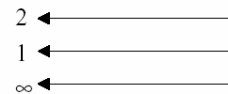
Exemple d'Unix

- 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
 - batteur => un processus à la fois
 - horloge => partagée par tous

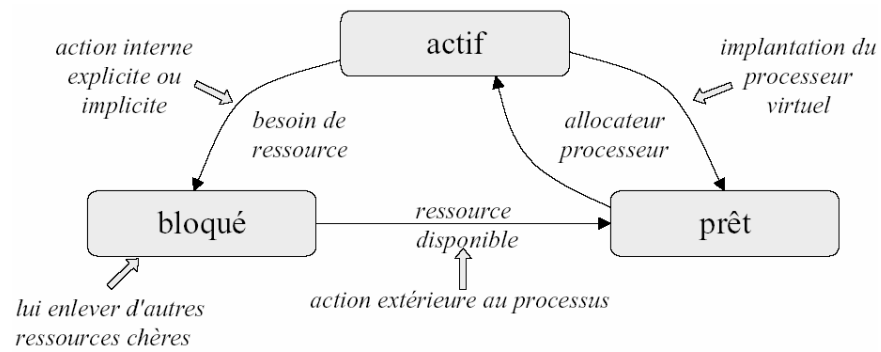


Ressource = toute entité dont a besoin un processus pour s'exécuter

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.ro.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

```
ostrom@trolliro.umontreal.ca /u/ostrom
troll 41 ~ostrom ps
      PID TTY          TIME CMD
      232923 ttyq1    0:00 tcsh
      232984 ttyq1    0:00 ps
troll 42 ~ostrom ps -ef
      UID          PID    PPID  C    STIME TTY          TIME CMD
      root           1         0  0   Oct 17 ?        1:37 /etc/init
      root          21         1  0   Oct 17 ?        0:00 /sbin/xlv_plexd -m 4 -w
1
      root          25         1  0   Oct 17 ?        0:00 /sbin/xlv_labd -v
      root          39        25  0   Oct 17 ?        0:00 /sbin/xlv_labd -v
      root          97         1  0   Oct 17 ?        0:01 /usr/etc/syslogd
      root         199         1  0   Oct 17 ?        0:00 /usr/etc/portmap -v -a 2
55.255.252.0,132.204.20.0 -a 255.255.252.0,132.204.24.0
      root         208         1  0   Oct 17 ?        0:14 /usr/etc/nsd -a nis_secu
rity=local
      root         210         1  0   Oct 17 ?        0:00 bio3d
      root         224         1  0   Oct 17 ?        0:00 /usr/etc/nfsd
      root         225        224  0   Oct 17 ?        0:00 /usr/etc/nfsd
      root         226        224  0   Oct 17 ?        0:02 /usr/etc/nfsd
      root         227        224  0   Oct 17 ?        0:00 /usr/etc/nfsd
      root         228        224  0   Oct 17 ?        0:00 /usr/etc/nfsd
      root         229        224  0   Oct 17 ?        0:00 /usr/etc/nfsd
      root         230        224  0   Oct 17 ?        0:02 /usr/etc/nfsd
```

Exemples: top

```
ostrom@trolliro.umontreal.ca /u/ostrom
IRIX64 troll 6.5 IP35          load averages: 0.00 0.00 0.00          23:26:51
54 processes: 53 sleeping, 1 running
16 CPUs: 99.8% idle, 0.0% usr, 0.2% ker, 0.0% wait, 0.0% xbrk, 0.0% intr
Memory: 16G max, 16G avail, 15G free, 1014M swap, 1014M free swap

      PID      PCRP USERNAME PRI  SIZE  RES STATE  TIME  WCPU% CPU% COMMAND
      233005    233005 ostrom   20 2080K 1296K run/3   0:00   0.1   0.23 top
      278        278 root    20 1568K  864K sleep  1:11   0.0   0.02 rwhod
      232952    152318 ostrom   20 4880K 3376K sleep  0:00   0.0   0.01 xterm
      320        320 root    32 1920K 1920K sleep  0:34   0.0   0.01 xntpd
      271        271 root    20 1728K 1072K sleep  0:01   0.0   0.01 inetd
```

Exemples: top

```

ostrom@trolliro.umontreal.ca /u/ostrom
IRIX64 troll 6.5 IP35          load averages: 0.88 0.35 0.13          23:31:45
64 processes: 62 sleeping, 2 running
16 CPUs: 93.6% idle, 6.3% usr, 0.2% ker, 0.0% wait, 0.0% xbrk, 0.0% intr
Memory: 16G max, 16G avail, 15G free, 1014M swap, 1014M free swap

  PID  PCRP  USERNAME  PRI  SIZE  RES  STATE  TIME  WCPU%  CPU%  COMMAND
 232564 232564  ostrom    16   26M   17M  run/8   2:10  98.4  99.83  MathKer
 232964 232964  ostrom    20 2096K 1296K  run/1   0:00   0.3   0.26   top
 232554 232554  ostrom    16 2832K 1616K  sleep   0:00   0.1   0.09   TIFF.ex
 232952 152318  ostrom    20 4880K 3376K  sleep   0:00   0.0   0.01   xterm
    278    278   root     20 1568K  864K  sleep   1:11   0.0   0.01   rwhod
    320    320   root     32 1920K 1920K  sleep   0:34   0.0   0.01   xntpd
  
```

Conclusions

- Processus = entité dynamique résultant des instructions
 - programme = description statique* →
 - processeur = organe d'exécution* →
 - *état mémoire ou dans les registres*
- 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-crée.
- + 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.