Les principaux Aspects des Systèmes d'Exploitation

N.Hameurlain
http://www.univ-pau.fr/~hameur

Plan

- Processus et synchronisation
- Algorithmique des processus
- Allocation de ressources
- Structuration des systèmes
- Gestion des objets

Processus et synchronisation

- Problèmes: identifier et définir les activités autonomes;
- Processus: activité résultant de l'exécution d'un ou plusieurs programmes séquentiellement en vue de la réalisation d'une tâche bien définie;
- synchronisation: coordination d 'activités autonomes mais dépendantes

Processus

- Le processus est l'entité d'affectation du processeur : en dehors des traitements d'interruptions et des périodes de transition, le processeur est alloué à un processus.
- Le mécanisme d'allocation du processeur (scheduler) considère les processus comme des activités autonomes et a priori indépendantes.

Synchronisation

- Problème: Comment faire pour qu'une activité A, arrivée à un point X de son exécution, attende pour continuer qu'une activité B ait atteint ou dépassé un point Y de son exécution ?
- Solutions: approche par temporisation , et par « synchronisation »

Approche par temporisation

- Principe: mesure le délai maximum D qu'il faut à B pour atteindre Y. Il suffit alors de commencer à exécuter A à partir de (début de B) + D;
- problème: solution difficile et périlleuse.

Approche par temporisation: problèmes

- Aléa des temps d'entrées/sorties et indéterminisme induit;
- Variabilité du [Delta] avec la charge;
- Pannes partielles;
- Modification des programmes si évolutions des délais;
- Mauvaise utilisation des ressources car utilisation d'un délai maximum.

Approche par synchronisation

- Conception de mécanismes de synchronisation utilisables par les processus qui doivent se synchroniser :
  - un processus signale aux autres processus un "événement" (changement significatif) par rapport à sa situation ou dans l'environnement d'exécution.
- Exemple : sémaphores (Dijkstra 1967), nombreuses autres propositions.

Mécanismes de synchronisation

- Doivent permettre
  - l'auto-suspension d'un processus
  - l'activation d'un processus suspendu
- Interaction forte avec le coeur du système d'exploitation car touche à l'allocation ou à l'abandon du processeur par un processus .

Développements actuels

- Intégration des mécanismes de synchronisation dans les langages de programmation de haut niveau;
- Conception d'outils de synchronisation pour les systèmes répartis;
- Méthodes de preuves de la validité des solutions aux problèmes de synchronisation.
- Processus légers (lightweight process ou thread)

Processus légers (Threads)

- Systèmes classiques (dérivés d'UNIX) : les processus ne peuvent communiquer que par des mécanismes impliquant des appels systèmes, ce qui est relativement long.
- Définition de plusieurs activités (threads) coopérantes au sein d'un même espace d'adresses de processus.
  - Communication, synchronisation inter-activités sans passer par le système.

Algorithmique des Processus(1)

- Quels sont les mécanismes que doit offrir un environnement d ’exécution de programmes pour:
  - permettre à un programmeur de concevoir une application, et
  - utiliser un ensemble de processus synchronisés?

Algorithmique des Processus (2)

- Déclaration de processus
- Activation d'un ou plusieurs processus
- Suspension d'un processus
- Réactivation d'un processus
- Etats d'un processus

Déclaration de Processus

Processus <*nom_proc*> (paramètres);
<déclaration des variables privées du processus>
début
<corps du processus>
fin

- Sémantique: un processus de nom <*nom_proc*> avec des paramètres et des variables privées:
  - permet d 'avoir un modèle de processus qui pourra ensuite être instancié plusieurs fois

Activation d'un processus (1)

<*nom_proc*> (paramètres)

L'activation ressemble à un appel de procédure, mais sa sémantique est très différente:

- La suite d 'instructions du processus activé est exécutée en parallèle avec la suite d 'instructions de l'instance appelante.
- Il n 'y a donc pas de retour de processus comme il y a un retour de procédure.
- Exemple UNIX: *Fork*, permet de lancer l 'exécution d 'un processus; et *Wait*: permet d 'attendre la fin d 'un quelconque des processus lancés précédemment par un *Fork*.

Activation d'un processus (2)

- Exécution séquentielle:
  - S(a, b) : a et b s 'exécutent en séquentiel, c-à-d, b s'exécute après a.
- Exécution Parallèle:
  - P(a, b): a et b s'exécutent en parallèle;
    - ils sont lancés en // sans faire aucune hypothèse sur leurs vitesses d 'exécution relatives;
    - La suite du programme n 'est exécutée que lorsque cette instruction est terminée (c-à-d tous les processus lancés sont terminés)

Suspension d'un processus(1)

- Permet à un programmeur de suspendre provisoirement l 'exécution d 'un processus lorsqu'il ne peut plus continuer;
  - Peut prendre la forme d'un appel de procédure dont l'effet est de suspendre l 'exécution du processus qui exécute cette procédure.
- L'environnement d 'exécution doit choisir un autre Processus à exécuter sur le processeur

Suspension d'un processus(2)

Suspendre (identificateur_d 'attente)

- Sémantique: le processus qui exécute un suspendre est provisoirement arrêté;
- la procédure suspendre ne se terminera que lorsque le processus sera réactivé ultérieurement;
- l'identificateur permet à l 'environnement d 'exécution de distinguer et de gérer plusieurs ensembles de processus suspendus

Réactivation d’un processus (1)

Réveiller (identificateur_d ’attente)

- Peut prendre la forme d’une procédure exécutée par le processus « réactiveur »:
  - La procédure se termine immédiatement.
- Sémantique: les processus suspendus à *identificateur_d ’attente* redeviennent activables et pourront être activés dès qu’un processeur sera disponible

Réactivation d’un processus (2)

- A la suite d ’un *Réveiller*, l ’environnement d’exécution peut choisir en toute liberté le prochain processus à exécuter parmi:
  - le processus « réactiveur »,
  - les processus réactivés,
  - et les autres processus.
- Le paramètre *identificateur_d ’attente* ne représente aucun objet particulier (variable, condition d ’attente ou autre)

Etats d’un processus

- *Actif*: il s ’exécute sur un processus;
- *Suspendu*: il ne peut progresser car une condition d ’attente est vraie;
- *Prêt*: il pourrait progresser mais aucun processeur ne lui est alloué pour s ’exécuter.

Allocation de ressources(1)

- Aspects qualitatifs et quantitatifs
- Problèmes en univers centralisé
  - gestion des processeurs,
  - gestions de l'espace disque,
  - gestion de la mémoire centrale,
  - traitement de l'interblocage.

Allocation de ressources(2)

- Problèmes en univers réparti:
  - gestion des voies communes;
  - contrôle des flux, contrôle de congestion dans les réseaux;
- Développement des techniques de simulation et modélisation mathématiques par systèmes de files d'attente

Structuration des systèmes

- Monolitiques;
- A couches;
- Client/Serveurs ou Micro-noyau.

Structuration des systèmes: système monolitique (1)

- Collection de procédures qui collaborent à la réalisation de fonctions complexes.
- Chaque procédure est visible a toutes les autres, et peut appeler a tout moment n'importe quelle autre;
- Exemple: UNIX

Système monolitique (2)

- Les services (Appels système) offerts par le SE sont adressées en :
  - mettant les paramètres dans des registres ou la pile; et en exécutant une instruction spéciale nommée appel au noyau ou appel au superviseur;
  - Le SE examine alors les paramètres pour déterminer l'appel système a exécuter (pointeur sur l'appel système)

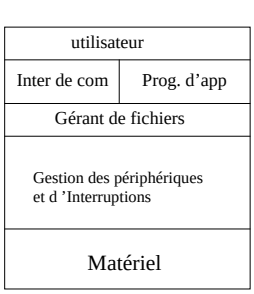
Système monolitique (3)

- Le SE appelle la procédure correspondante (localisation et identification du service, qui est ensuite appelé);
- Le contrôle est rendu au programme de l'utilisateur;

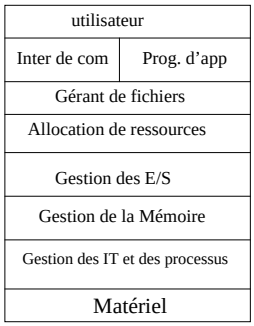
Systèmes à couches(1)

- Vu comme un ensemble de couches logiciels superposées.
- Chaque couche utilise les couches inférieures et fournis un service supplémentaire aux couches supérieures
- Exemple: THE, MULTICS,..

Systèmes à couches(2)



Mono-utilisateur monotâche



Idéal

Système Client/Serveur ou Micro-noyau(1)

- Certaines fonctions complexes du système sont sorties du système proprement dit et confiées à des serveurs indépendants (serveur de fichiers, de mémoire, etc…)
- Les processus utilisateur (clients) et les serveurs s 'exécutent en mode utilisateur;
- La communication entre clients et serveurs se fait par envoi de messages à travers le noyau du système (mode noyau);

Système micro-noyau (2)

- Système configurable: changer les serveurs pour obtenir une meilleur adéquation;
- Système transportable: il suffit de réécrire la couche micro-noyau qui coordonne tous les serveurs;
- Incorporation des systèmes existants(DOS, UNIX)
- Se prête bien à une utilisation sur les systèmes distribués(Amoeba, Mach, …).

Gestion des Objets (1)

- Un système d'exploitation gère des objets (structures d'informations) comme le fait un langage de programmation et définit les moyens d'accès à ces objets.
- Exemples : fichiers, ..

Gestion des Objets (2)

- Problèmes de gestion des objets
  - implantation ; désignation
  - protection ;partage; destruction
  - migration ; communication;
  - réutilisation
- Définition de serveurs d'objets normalisés : CORBA