Multi-tâches et multi-threadé

RETOUR

Multitâche: Plusieurs applications peuvent être lancées et exécutées simultanément, qu'elles soient relatives au système d'exploitation ou qu'elles aient été lancées par les utilisateurs. Pas d'exécution véritablement parallèle dans le cas des machines mono-processeur.
L'ordinateur n'exécute qu'une seule application (une seule instruction processeur) à tout instant, mais switche de manière très rapide entre toutes les applications en exécution par affectation successive de l'utilisation exclusive du processeur. Le cadencement étant très rapide, l'impression est ainsi donnée que les applications s'exécutent en parallèle.

Deux types de multi-tâches:

- Multitâche préemptif: Le système d'exploitation gère l'attribution des temps de calcul aux applications sans que celles-ci aient à s'occuper de quoi que ce soit (elles ne peuvent même-pas l'empêcher). Toute application peut fonctionner en multitâche dans ce cadre.
- Multitâche non-préemptif: Les applications rendent elles-mêmes la main au système d'exploitation et gèrent leurs contextes d'exécution respectifs. Les applications doivent être conçues spécifiquement dans ce cadre, sinon le multitâche ne fonctionne pas.

Gestion du multitâche préemptif au sein de Windows (comme dans le cas de tous les systèmes d'exploitation modernes).

Service : Application généralement liée au système d'exploitation fonctionnant en tâche de fond avec ou sans action à l'écran ("démon" dans la terminologie Unix).

Thread : Un développeur peut concevoir une application sous la forme d'un programme maître P créant lui-même des sous-programmes SP_i identiques ou non s'exécutant indépendamment et "parallèlement" au sein de la zone mémoire associée à P : des threads (processus léger).
Outre un intérêt algorithmique, la programmation multi-

threadée a pour caractéristique que, si un processeur multicoeur est utilisé et que le système d'exploitation gère cette fonctionnalité, les différents threads seront affectés à différents coeur, réalisant ainsi une véritable exécution parallèle.
Un autre intérêt de la programmation multi-threadée est que le processus père et ses fils partagent la même zone mémoire facilitant ainsi le partage d'informations entre eux (aucun contrôle, aucune impossibilité de communication).
Ceci peut aussi être un inconvénient car le système ne gère aucune protection des données -> possibilité d'anarchie mémoire et donc de plantage.

Existence de priorités attribuées par le système d'exploitation (modifiables par l'administrateur) pour gérer l'ordonnancement de l'attribution de temps CPU aux applications et à leurs threads. Ces priorités sont primordiales pour le bon fonctionnement du système d'exploitation.