

Chapitre 11 L'EXECUTIF DE WINDOWS NT

Objets, processus et gestion de la mémoire

Comme il a été indiqué au chapitre 9, l'exécutif de WINDOWS NT met des sous-services standard à la disposition des applications clientes, gère les processus et les tâches (threads), gère la mémoire et les E/S.

Les ressources système sont représentées par des objets, mais WINDOWS NT est écrit en C de Microsoft et ne dispose pas vraiment de structures orientées objet.

1. GESTION DES OBJETS

Les objets de l'exécutif permettent trois fonctionnalités :

- offrir des noms explicites pour les ressources du système
- partager ressources et données entre les processus
- protéger les ressources

Seules les structures concernées par ces traitements sont modélisées par des objets.

Il existe deux sortes d'objets :

- **les objets de l'exécutif** : accessibles par les sous-systèmes protégés en mode utilisateur (processus, tâche, jeton d'accès, sémaphore, fichier).

- **les objets du noyau** : accessibles par les couches basses du système (interruption, tâche-noyau, processus-noyau, etc...)

Définition : on appelle *handle d'objet* un index dans une table de pointeurs sur les objets spécifique à un processus. Cette table comprend des pointeurs vers tous les objets pour lesquels le processus a ouvert un *handle*. Un *handle* est donc un pointeur indirect vers un objet. Seul le gestionnaire d'objets a le droit de créer un *handle*.

Pour un processus, un *handle* d'objets est un pointeur indirect sur des ressources du système. On évite ainsi que les applications manipulent directement les structures de données du système.

1.1 Structure des objets

Un objet comporte deux parties :

- **un en-tête** contrôlé par le gestionnaire d'objets. Les services génériques (fermer un handle, dupliquer un handle, interroger l'objet, interroger la sécurité, changer la sécurité, attendre un objet) accèdent aux données de l'en-tête.

- **un corps** contrôlé par le sous-ensemble du SE qui a créé ce type d'objet. Le corps comprend plusieurs champs : type d'objet, attributs de l'objet, services de l'objet.

1.2 Gestionnaire d'objets

Il assure des fonctionnalités nombreuses :

- vérification des droits d'accès des tâches et processus sur les objets et réalisation de leur protection

- création et suppression des objets; surveillance des objets temporaires pour les supprimer dès qu'ils ne sont plus utilisés; suivi des objets en utilisant leur en-tête

- centralisation des opérations de contrôle des ressources

- attribution des noms aux objets (chaque nom est global sur une machine, mais pas dans le réseau) et implantation des objets nommés dans une mémoire globale gérée dynamiquement, attribution d'alias aux noms. Cet espace est structuré en arbre.

2. LES PROCESSUS ET LES TÂCHES (THREADS)

Chaque processus est découpé en une ou plusieurs unités d'exécution ou tâches ou *threads*. Le découpage en *threads* permet à un processus de paralléliser l'exécution d'une partie de son code ou d'une partie de celui-ci.

Processus et *threads* sont implémentés sous formes d'objets. Le gestionnaire de processus de l'exécutif crée ou supprime des processus et des *threads* et gère les communications interprocessus. Le gestionnaire de processus fournit des services pour la gestion des processus liés aux sous-systèmes d'environnement. Il n'y a pas de relation père-fils entre les processus natifs de WINDOWS NT.

2.1 Les processus

Un processus WINDOWS NT comprend :

- un programme exécutable (code et données)
- un espace d'adressage virtuel *privé*
- un ou plusieurs threads auxquels des ressources système sont affectées (sémaphores, fichiers, ports, ...)

La plupart des processus fonctionnent en mode utilisateur : ainsi, leur espace d'adressage est protégé des applications et des autres sous-systèmes.

A chaque processus est associé un ensemble de ressources : jeton d'accès, table d'objets, structure de données pour le gestionnaire de la mémoire virtuelle

La structure de processus de l'exécutif NT fournit des mécanismes de base qui permettent aux sous-systèmes d'environnement (POSIX, Win32, OS/2) de construire leur propre structure de processus et de la gérer.

2.2 Les tâches

Un thread appartient à un processus et à un seul. Sa structure comporte les éléments du contexte:

- un compteur ordinal
- une pile en mode utilisateur
- une pile en mode noyau
- une zone de données privées
- un mot d'état du processeur

Tous les threads d'un processus peuvent accéder à son espace d'adressage, aux handles d'objets et à ses autres ressources.

Lorsqu'un thread en mode utilisateur fait appel au SE, il est basculé en mode noyau et le service demandé est exécuté après vérification de ses paramètres. Puis, le thread est basculé à nouveau en mode utilisateur.

Il existe deux moyens de communications entre threads :

- les **objets de synchronisation** qui implémentent des fonctions d'attente et de signalisation. Un objet de synchronisation possède deux états : signalé et non signalé

- la **fonction d'alerte**, notifiée par l'APC (Asynchronous Procedure Call) permet d'interrompre l'exécution d'un thread pour lui faire exécuter une procédure (à comparer aux signaux d'UNIX)

3. COMMUNICATION INTER-PROCESSUS

Il existe 3 moyens de faire communiquer des processus avec WINDOWS NT : les appels de procédure locale, les appels de procédure distante et les canaux nommés.

3.1 LPC, Local Procedure Call

C'est un mécanisme rapide de transmission de message entre processus situés sur la même machine.

Le processus client envoie une demande de connexion sur le port de connexion du processus serveur. Le processus serveur crée deux objets-ports de communication pour établir un canal entre client et serveur et retourne au client l'un des handles correspondant.

A la mise en place du canal de communication, le processus client indique la méthode LPC qu'il veut utiliser parmi les trois suivantes :

- **envoi d'un message dans la file de message** de l'objet-port du processus serveur (1 bloc de message = 256 octets). Cette méthode, valable pour des messages courts, relie les espaces d'adressage des processus client et serveur par l'intermédiaire de la mémoire système allouée à l'objet-port

- **envoi, au port du serveur, d'un pointeur sur le message et passage du message en mémoire partagée**. Le processus client crée un objet de mémoire partagée : objet-section associé au port de communication du client, mais visible des deux processus. Lorsque le processus serveur répond, il crée aussi un objet-section associé à son propre port de communication

- **transmission d'un message à un thread spécifique d'un serveur à travers une mémoire partagée**. Ce mécanisme est appelé "LPC rapide". Il est utilisé par Win32 pour augmenter ses performances dans la gestion des fenêtres et des graphismes. Dès qu'un processus serveur reçoit une demande de transmission LPC rapide de messages sur son port de connexion, Win32 crée trois ressources pour le client :

- * un thread pour le traitement des requêtes
- * un objet-section de 64 Ko de mémoire partagée pour la transmission des messages
- * un objet-"paire d'événements" pour synchroniser le thread client et le thread serveur

3.2 RPC, Remote Procedure Call

L'objectif est de permettre à une application distribuée sur plusieurs machines d'appeler des services disponibles sur diverses machines du réseau. Ce mécanisme est conforme aux standards RPC de l'OSF.

Pour traiter des procédures distantes, une application utilise une bibliothèque locale de liaison dynamique (Dynamic Link Library ou **DLL**), comprenant des procédures relais qui portent le même nom que les procédures distantes et possèdent les mêmes interfaces.

Le relais récupère une copie des données pointées par un de ses paramètres, résout les références et appelle des procédures d'exécution de RPC qui s'adressent à la machine distante.

3.3 Les canaux nommés

Ce mécanisme fait abstraction des problèmes de routage et de transmission de données dans le réseau. Il est implémenté par un pseudo-SGF : le pilote du système de fichiers des canaux nommés.

Chaque canal nommé est implémenté par un objet-fichier et dispose des mêmes mécanismes de sécurité que les autres objets de l'exécutif. Un canal nommé est utile pour l'envoi de flots de données entre client et serveur d'une application distribuée.

4. Gestion des processus et threads par le noyau

Le noyau de WINDOWS NT est écrit en C de Microsoft et en Assembleur. Il assure le contrôle des processeurs et fournit des fonctions de bas niveau à l'exécutif NT. Pour l'essentiel, le noyau assure l'ordonnancement des processus et des threads, et gère les interruptions.

Le code du noyau n'est pas paginé hors de la mémoire et n'est pas interruptible. Il s'exécute toujours en mode noyau.

Le rôle du noyau peut se résumer à **4** éléments principaux :

4.1 Ordonnancement des threads

L'ordonnanceur des tâches est appelé "*module répartiteur du noyau*". Chaque tâche peut se trouver dans l'un des 6 états suivants :

- **prêt** : en attente d'exécution
- **standby** : un thread et un seul est dans cet état à un instant donné sur un processeur donné; c'est le thread dont l'exécution sera lancée à la prochaine commutation de tâches
- **en exécution** : état du thread en cours d'exécution. Si le noyau l'interrompt pour exécuter un thread de priorité plus élevée, il est placé en tête de la file d'attente, dans l'état **prêt**
- **attente** : le thread attend qu'un objet soit disponible ("mis en état signalé") . Le noyau l'informerá que le répartiteur rendra disponible cet objet attendu
- **transitoire** : le thread pourrait être prêt, mais une ressource est indisponible. Dès que la ressource attendue est disponible, il passe à l'état **prêt**
- **terminé** : son exécution est achevée. Il est supprimé ou bien réinitialisé.

Les niveaux de priorité des threads sont numérotés de 1 (le plus bas niveau) à 31 (le plus élevé). Les niveaux 16 à 31 sont réservés aux threads temps réel et les niveaux 1 à 15 sont alloués aux threads à priorité variable. L'ordonnanceur gère une file d'attente par niveau de priorité.

4.2 Traitement des interruptions

Lorsque survient une interruption, le processeur transfère le contrôle au code de traitement des déroutements du noyau. Celui-ci transfère à son tour le contrôle au module chargé du traitement de l'interruption qui a été décelée.

Le noyau utilise un ensemble standard de priorités d'interruptions (Interrupt Request Level ou **IRQL**), portables d'un processeur à un autre.

Les exceptions (erreurs **synchrones** résultant de l'exécution d'une instruction en langage machine) sont traitées en mode utilisateur ou en mode noyau selon un mécanisme analogue.

4.3 Synchronisation des processeurs

Sur une machine multiprocesseurs, le noyau assure la synchronisation des processeurs. Il dispose de primitives d'exclusion mutuelle pour la gestion de l'accès aux données globales à tous les processeurs (mécanisme dit de **verrou continu**).

Le noyau fournit aussi des mécanismes de synchronisation entre processeurs appelés **objets-répartiteurs**.

4.4 Reprise après interruption d'alimentation

Le noyau réserve à la coupure d'alimentation le deuxième niveau d'interruption par ordre décroissant (le niveau le plus élevé est alloué aux erreurs de bus et aux tests défectueux de la machine). L'objectif est de permettre la reprise des opérations d'E/S pour tout ordinateur muni d'une batterie de secours pour la mémoire, après une coupure d'alimentation. deux objets sont dédiés à la gestion de la reprise : objet-notification d'alimentation et objet-état d'alimentation.

5. GESTION DES ENTREES-SORTIES

Le système d'E/S de WINDOWS NT offre un accès aux périphériques sous forme de fichiers virtuels, manipulés par des *handles* de fichiers. Tous les pilotes de périphériques communiquent à l'aide de paquets de demandes d'E/S (Input/Output Request Packet ou **IRP**). Tous les périphériques physiques, sources ou destinations, du système d'E/S sont modélisés par des **objets-fichier**. Dès le chargement d'un pilote de périphérique, l'objet-pilote associé et l'objet-périphérique associé permettent de traiter toute demande d'accès en occultant au maximum les caractéristiques physiques du périphérique.

Pour chaque opération d'E/S, le système d'E/S crée un IRP, le passe au pilote qui doit le traiter et récupère l'IRP lorsque l'E/S est achevée.

Les périphériques courants : clavier, souris, écran, imprimante, sont gérés par un **pilote monocouche**. Les unités de disques et les gestionnaires de réseaux ont des **pilotes multicouches** (pilote de fichiers, puis pilotes de périphériques).

Les principales caractéristiques du système d'E/S sont les suivantes :

- il admet plusieurs systèmes de fichiers : celui de MS-DOS, celui d'OS/2, CDFS (système de fichiers de CD-ROM), et bien sûr NTFS (celui de Windows NT)

- on peut ajouter ou enlever dynamiquement des pilotes; ceux-ci ont une structure uniforme et modulaire

- il protège ses ressources partageables par l'utilisation d'objets

- il supporte les interfaces d'E/S de Win32, OS/2 et POSIX

BIBLIOGRAPHIE SUCCINCTE

Helen CUSTER Au cœur de Windows NT, Microsoft Press, 1993

Bruno DARDONVILLE Architecture de Windows NT, Hermès, 1996

Paul YAO An Introduction to Windows NT Memory Management Fundamentals,
Microsoft Development Library, 1992