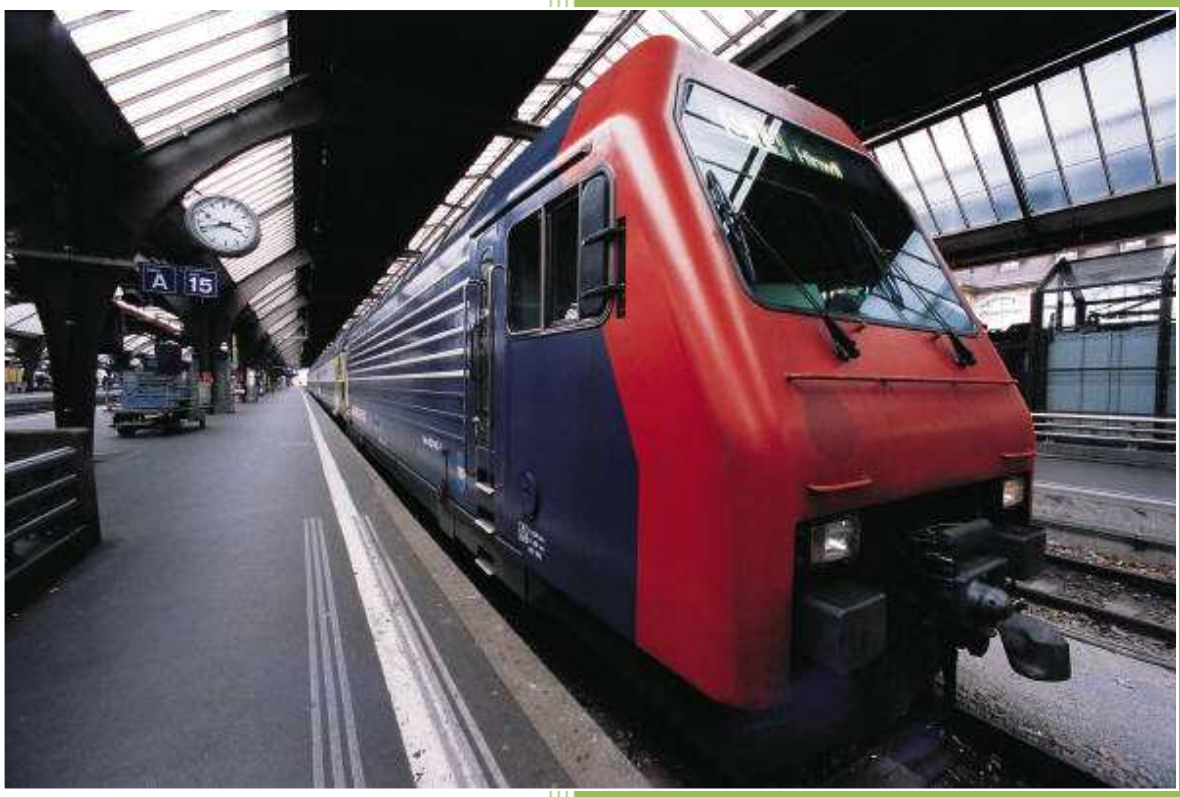
2008

Présentation de Windows NT



Présentation de Windows NT

Sommaire

1. La stratégie de Windows NT

Dans ce chapitre nous exposons un point de vue général sur la conception de Windows NT. Ce système est basé sur une architecture utilisant un micro-noyau, comme Chorus. Par contre, à l'opposé de Chorus, Windows NT n'est pas un système distribué. Tous les accès au réseau se font de manière explicite.

1.1 Organisation en trois couches

L'architecture de Windows NT est organisée en trois couches distinctes : la couche Hardware Abstraction Layer (HAL), la couche exécutive et la couche des applications. Une couche ne communique qu'avec ses couches adjacentes. Nous pouvons remarquer que la couche application supporte plusieurs applications (essentiel pour un système multitâche).

La couche HAL, comme son nom l'indique, a pour but de masquer le matériel utilisé. De cette manière, la couche exécutive voit la même machine, que le système tourne sur un processeur PowerPC ou sur un Intel. La couche exécutive doit, en utilisant les services de la couche HAL, rendre les services qu'offre la plupart des systèmes d'exploitation : gestion des processus, gestion des ressources, ...

Par conséquent, la couche HALest la seule à devoir être portée lors de la mise en place du système sur un autre type de machine. La couche exécutive utilisant uniquement les directives de la couche HAL, elle ne nécessite aucun changement lors du portage du système.

1.2 Les différentes API offertes

Les concepteurs de Windows NT avaient pour but de permettre l'exécution, non seulement des applications Windows et MsDos, mais en plus d'applications provenant de différents systèmes déjà existants : POSIX et OS/2. Pour cela, il fallait respecter les différentes API de ces systèmes. La création d'une seule API pour la couche executive nécessite l'encapsulation de toutes les autres API. Une telle architecture est trop complexe et les performances du système seraient insuffisantes.

Une autre solution a donc été adoptée : la création de sous-systèmes dans la couche des applications. Les applications sont clientes du sous-système caractérisant leur système natif. Un sous-système met à la disposition des applications une API identique à celle du système natif. Les sous-systèmes sont de deux types différents : les sous-systèmes d'environnement et les sous-systèmes intégraux.

Les sous-systèmes intégraux sont des serveurs en mode utilisateur qui exécutent des fonctions importantes du système. Par exemple, le sous-système de sécurité surveille les accès aux ressources systèmes et gère une base de données des comptes utilisateurs (noms, mots de passe, ...).

Un sous-système d'environnement est un serveur en mode utilisateur qui " fournit " une API spécifique. Le sous-système d'environnement natif Windows NT est le sous-système Win32.

Il est impossible à une application de demander les services de plusieurs API. Ceci n'aurait de toute manière aucun sens. En effet, la notion de handler de fichier au sens POSIX n'est pas la même que celle donnée par l'API Win32. Par exemple, les systèmes d'exploitation émulés par les sous-systèmes POSIX et OS/2 n'ont aucune compatibilité. Microsoft a décidé de changer l'API de Windows et MsDos pour passer à une API 32 bits. Les " systèmes " MsDos et Windows étant des systèmes 16 bits, Microsoft a décidé de ne plus continuer avec ces API mais d'en créer une nouvelle qu'ils ont appelée API Win32. Cependant, Windows NT devant exécuter les applications de MsDos et de Windows, un " émulateur ", désigné par l'acronyme Virtual Dos Machine (VDM) a été élaboré. Les applications font appel aux services offerts par leur API. Nous avons vu ci-dessus qu'une application ne fait appel qu'à une seule API. Ce sont les sous-systèmes qui font appel aux services de la couche exécutive, ceux-ci sont désignés par le terme services natifs.

Les applications entrent en communication avec les sous-systèmes grâce à des messages qu'ils s'échangent à travers la couche exécutive : les LPC (Local Procédure Call). Par conséquent, une

application désirant faire une entrée/sortie doit utiliser un LPC pour communiquer avec son sous-système. C'est le sous-système qui effectue l'appel au service d'entrée/sortie de la couche exécutive.

Le fonctionnement des LPC est basé sur un modèle client/serveur tout comme les RPC. Les applications sont des clients des sous-systèmes (serveurs). Les LPC assurent la cohérence du modèle client/serveur.

1.3 Les LPC

Les LPC sont une version localement optimisée d'un mécanisme de transmission de messages, plus généralement appelé RPC. Les LPC correspondent aux RPC/OSF (non compatibles avec les RPC SUN). L'utilité de telles procédures est de pouvoir communiquer entre processus. En effet, les processus de Windows NT possèdent chacun leur propre espace d'adressage, il est donc impossible d'appeler directement une procédure d'un autre processus. Les LPC sont des procédures qui font appel à des services d'un autre processus.

1.3.1 Principe

La communication entre les processus s'effectue d'une manière similaire à celle utilisée avec Chorus. C'est-à-dire que le processus serveur reçoit une requête de connexion sur son port. Il crée alors un nouveau port pour communiquer avec l'appelant. En fait, le serveur crée les deux ports de communication et retourne le port du client.

Le paragraphe suivant présente comment sont réalisées les Local Procedure Call dans Windows NT.

1.3.2 Réalisations

Il existe trois réalisations des LPC. Elles utilisent les ports associés aux processus. En effet, chaque processus est équipé d'au moins un port, au sens Chorus, et peut donc être contacté par ce biais.

La première et la plus simple des réalisations est utilisée pour les messages ne nécessitant que peu de place mémoire. En effet, cette réalisation consiste à envoyer les requêtes et recevoir les réponses à travers les ports respectifs des processus. Par contre, le port associé à chaque processus ne peut recevoir qu'un nombre limité de messages. De plus, la taille de ces messages est limitée à 256 octets, ce qui restreint son champ d'application.

La deuxième réalisation consiste à partager une zone mémoire chez le client, à y inscrire les messages et à prévenir le serveur. En effet, un message contenant l'adresse des données, est envoyé sur le port du serveur. Cette technique est utilisée pour les messages de plus de 256 octets.

La troisième réalisation a été implantée pour le sous-système Win32. L'envoi de messages entre processus est très coûteux en appels systèmes car il nécessite des recopies de messages de l'espace d'adressage de l'appelant vers l'espace d'adressage de l'appelé. Pour éviter cela, les concepteurs du sous-système Win32 ont mis en place un protocole appelé LPC rapide. Ce protocole est simple, il utilise un LPC classique d'initialisation. Le sous-système crée alors une zone de mémoire partagée, un objet événement paire et un thread dédié à l'échange. Les échanges de messages se font au travers de la mémoire partagée. Dans la deuxième solution, la synchronisation entre le client et le serveur (respectivement une application et un sous-système) est réalisée avec des messages. Dans cette solution, la synchronisation est effectuée grâce aux objets " événement pair ". Un événement pair est un objet de synchronisation entre deux processus. Cet événement est semblable à une bascule : lorsque le serveur active l'événement paire, il réveille le client et inversement lorsque le client le positionne.

1.3.3 Optimisations

Les applications utilisant l'API Win32 étant nombreuses, certaines optimisations ont été réalisées sur les LPC à destination du sous-système Win32 :

- Lorsqu'un objet est créé, ses propriétés sont recopiées dans la DLL cliente. En effet, la probabilité d'utiliser un objet auquel on vient d'accéder est forte et par conséquent le recopier dans la DLL cliente évite de devoir y accéder à travers un nouvel LPC.
- Lors d'utilisation de LPC concernant l'affichage, ceux-ci sont cumulés avant de les envoyer tous ensembles. Un LPC nécessite pour sa réalisation un changement de contexte. Par conséquent, un processus désirant effectuer un LPC doit attendre que le sous-système soit choisi par l'ordonnanceur pour que ce dernier effectue sa requête et lui réponde. Enfin, il reprend la main et obtient la réponse désirée. Lorsqu'il s'agit de requêtes graphiques, la réponse n'influe que rarement sur la suite du programme. La DLL cliente peut donc cumuler quelques requêtes graphiques avant de les envoyer. Ceci évite de devoir céder son quantum de temps à chaque affichage d'une ligne.
- L'API Win32 a été réalisée pour améliorer l'API MsDos. Par conséquent, elle hérite de la contrainte des noms de lecteurs logiques de MsDos. La concordance entre les lecteurs logiques de MsDos (A: ,B: ,C: ,...) et les objets de Windows NT devrait être assurée par le sous-système Win32. En fait, cette concordance est directement prise en charge par le noyau. L'accès à ces lecteurs étant fréquent, des alias ont été réalisés dans la couche exécutive. La couche exécutive répond alors immédiatement, les changements de contexte nécessaires pour que le sous-système puisse répondre sont évités.

1.4 La sécurité

Pour pouvoir pénétrer dans le système, un utilisateur doit s'identifier et s'authentifier. Ces actions peuvent se faire de diverses manières. La manière la plus connue est celle utilisée par la plupart des Unix, c'est-à-dire donner son nom (identification) et son mot de passe (authentification). D'autres solutions d'authentification sont envisageables telles que l'empreinte du pouce ou de la rétine de l'œil.

La procédure d'identification et d'authentification s'intitule procédure de logon. Cette procédure passe la main au système de sécurité qui vérifie si la personne est bien qui elle prétend être. Puis il crée un jeton d'accès (acces token) contenant les différentes informations de l'utilisateur telles que son nom, ses groupes, sa priorité et les droits par défaut de ses fichiers. Finalement, l'utilisateur obtient la main car le système de sécurité lance le shell par défaut de l'utilisateur, en lui passant le jeton d'accès.

Le jeton d'accès est la carte d'identité de l'utilisateur, il le suivra dans toutes les applications utilisées. La gestion de la sécurité nécessite la connaissance de diverses informations. Le jeton d'accès comporte ces données.

La discrétion et la confidentialité des objets que gère l'utilisateur, sont assurées par le système de sécurité. Celui-ci vérifie, lors de chaque action sur un objet, la légalité de l'action demandée. En effet, chaque objet possède un descripteur de sécurité qui donne les droits accordés aux différents utilisateurs. Il s'agit d'une liste dont les éléments comportent le nom d'un utilisateur ou d'un groupe et les droits qui lui sont associés. Cette liste est parcourue dans l'ordre. La première entrée concordante est utilisée.

Windows NT offre les quatre fonctionnalités suivantes :

a) *1. identification et authentification de l'utilisateur (logon),*

b) *2. gestion des droits d'accès des données utilisateurs avec une granularité de niveau utilisateur,*

c) *3. audit sur les actions effectuées,*

d) *4. réinitialisation de la mémoire avant chaque réutilisation de celle-ci.*

Windows NT est actuellement de classe C2. Il vise le niveau B2 de façon à respecter les contraintes de sécurité des applications militaires.

1.5 La répartition du système

Windows NT est un système multitâche et multiprocesseur. Il existe différents types de systèmes multiprocesseurs : les systèmes asymétriques et les systèmes symétriques.

Un système asymétrique est un système multiprocesseur qui exécute le système d'exploitation sur un processeur et les applications sur les autres.

Dans un système symétrique, il n'y a pas d'affinité de processeur. Le système d'exploitation comme les applications peuvent s'exécuter sur n'importe quel processeur. Cela signifie que les processeurs sont totalement banalisés. Windows NT est un système symétrique.

1.6 Une vision à l'échelle mondiale

Pour répondre à l'hétérogénéité des représentations de données, utilisées par les différents pays, Windows NT définit un format spécial des données, appelé Unicode. Ce format code les caractères sur 16 bits, ce qui représente un jeu de 65535 caractères. Windows NT utilise également un jeu de caractères, le Windows ANSI code set, qui enrichit l'ASCII (American Standard Code for Information Interchange) de tous les caractères accentués européens.

1.7 Architecture globale et méthode d'appel

Nous avons vu dans la partie 1.1 que Windows NT est organisé en trois couches. Nous avons expliqué dans la partie 1.2 l'utilisation de la couche application mais nous n'avons pas encore détaillé la couche exécutive. Cette couche est composée de divers modules ayant chacun une tâche bien précise.

Chaque module de la couche executive met à disposition ses services aux autres modules de la couche, ainsi qu'aux modules de la couche application. Les sous-systèmes utilisent les services natifs, ceux offerts par la couche executive.

Tout sous-système appelle directement les services systèmes de l'executive. Le sous-système POSIX est client du sous-système Win32. Mais il peut aussi appeler des fonctions systèmes de l'executive, avec l'objectif de minimiser la circulation de messages entre les applications POSIX, le sous-système POSIX et le sous-système Win32.

Chaque module est différent et manipule ses propres données systèmes. Par exemple, le gestionnaire de mémoire virtuelle ne peut consulter directement les données du gestionnaire de processus. Chaque module offre une interface soigneusement contrôlée aux autres modules. Ainsi, il est possible de construire de nouveaux modules dans l'exécutive en ne connaissant que les interfaces des autres modules.

1.8 Conclusion

Le modèle client/serveur est un élément fondamental de l'architecture de Windows NT. Ce modèle permet la communication entre les sous-systèmes et les applications. Les applications comme les sous-systèmes s'exécutent en mode utilisateur. Les sous-systèmes sont dits protégés. En effet, ils ne communiquent directement qu'avec le noyau. Le noyau assume le rôle de média de communication au sein du modèle client/serveur. Le noyau assure la cohérence des sous-systèmes.

2. Les objets

Dans cette partie, la notion d'objet dans Windows NT est détaillée. C'est l'implantation de l'exécutive qui a amené à la création des objets. Les concepteurs de l'exécutive NT ont choisi d'utiliser les objets pour représenter les ressources systèmes car ils permettent d'accomplir trois tâches importantes :

- Offrir des noms en clair,
- Partager les ressources et les données entre les processus,
- Protéger les ressources de tout accès non autorisé.

Toutes les structures de données de l'exécutive NT ne sont pas des objets, seules celles qui doivent être protégées, partagées, nommées ou rendues visibles à partir des applications en mode utilisateur sont placées dans des objets.

2.1 Deux types objets

Deux types d'objets se trouvent au sein de l'executive NT : les objets noyaux et les objets de l'executive.

Les objets noyaux ne sont pas visibles depuis les applications en mode utilisateur. Ils sont créés et utilisés uniquement au sein de l'executive NT. Ils ne sont mis en œuvre que par les couches les plus basses du système dans le noyau. Ils sont utilisés pour la communication entre modules.

Les objets de l'executive contiennent un ou plusieurs objets noyaux. Ils permettent de gérer les ressources systèmes et très partiellement la répartition. Ces objets nommés ont une portée limitée au système local, même si le système est multiprocesseur. Cette limitation va à l'encontre du principe de la distribution. Dans Chorus, l'UI d'un objet est global à un site.

Cependant, Windows NT propose un mécanisme de crochet, hook, permettant la gestion répartie des fichiers. Ce mécanisme n'est pour l'instant implanté que pour les fichiers.

2.2 Les objets de l'executive NT

Un système d'exploitation doit permettre d'utiliser différentes entités, tels que les imprimantes, les temps CPU (à travers les processus), les fichiers, ... Le système doit donc permettre de distinguer les différentes ressources disponibles et permettre à un utilisateur d'y accéder. Windows NT nomme tous ses objets. Ce nom permet à une application de retrouver facilement une ressource dont il a besoin.

La gestion des noms, du partage et de la protection des ressources représentées par des objets est centralisée (cette gestion n'en est que plus simple).

Les objets de Windows NT sont :

- les processus,

- les threads,
- les sections (mémoire partagée entre processus),
- les fichiers,
- les ports,
- les jetons d'accès,
- les objets événements,
- les sémaphores,
- les mutants (utilisés pour les exclusions mutuelles),
- les timers,
- les répertoires,
- les liens symboliques,
- les profils (mécanisme de mesure du temps d'exécution),
- les clés (référence d'un champ de la base de configuration de Windows NT).

Les objets ont une base commune, illustrée par le tableau ci-dessous, qui permet au gestionnaire d'objets de la couche exécutive de traiter tout objet indépendamment de sa signification réelle. Le gestionnaire d'objets permet :

- la distinction des objets,
- la recherche d'un objet particulier,
- le partage d'un objet entre plusieurs processus.

Attributs :

Object name	Nom de l'objet pour le rendre accessible
Object directory	Répertoire où l'objet se trouve (lié à la structure arborescente de Windows NT)
Security descriptor	Liste de contrôle d'accès de l'objet
Quota charges	Charge imputée à un processus qui ouvre cet objet
Open handle counter	Compteur du nombre d'ouvertures de l'objet
Open handler database	Liste des processus qui ont ouvert l'objet
Permanent/temporary status	Indicateur de pérennité de l'objet
Kernel/user mode	Indicateur de permission d'accès à l'objet en mode utilisateur
Type object pointer	" type de l'objet "

Procédures :

Close	fermeture de l'objet
Duplicate	partage de l'objet en dupliquant son handler
Query object	pour obtenir des informations sur l'objet
Query security	pour obtenir des informations sur les droits d'accès de l'objet
Set security	pour positionner les droits d'accès
Wait for single object	synchronise un thread avec un objet
Wait for multiple objects	synchronise un thread avec plusieurs objets

Le champ "open handler database" permet d'identifier les objets processus qui ont ouvert l'objet. C'est par ce biais que ceux-ci peuvent être avertis d'un changement de l'objet. Si nous prenons comme exemple un lecteur de CD-ROM, divers programmes ont ouvert un handlersur lui. Lorsque

l'utilisateur change de CD-ROM, le système est averti par une interruption matérielle, et il peut alors avertir les processus qui utilisent le lecteur. Cette manipulation serait impossible si le champ " open handler database " n'était pas présent.

Le gestionnaire d'objets est un élément de la couche executive. Il gère l'utilisation de tous les objets et vérifie la légalité des opérations effectuées. Il offre également un outil statistique d'utilisation des objets permettant l'audit de ceux-ci.

Tout objet de l'executive NT est manipulé par l'intermédiaire d'un handle. La création de l'objet ou l'ouverture d'un handle sur un objet nécessite l'utilisation explicite d'un nom. Le handle est un synonyme du couple <nom , objet>. Lors d'une référence à un objet, le gestionnaire d'objets n'a pas à rechercher l'objet concerné dans son espace des noms. En effet, le handle est une référence dans la table d'objets d'un processus. La " portée " du handle est donc limitée au processus.

Un processus obtient un handle :

- A la création d'un nouvel objet,
- A l'ouverture d'un objet existant,
- Par héritage d'un autre processus,
- Par duplication d'un handle existant dans un processus distant.

2.3 L'organisation des objets gérés par Windows NT

La gestion des objets touche également leur disposition les uns par rapport aux autres. Windows NT adopte la stratégie des systèmes d'exploitation UNIX à savoir l'organisation en arborescence. En effet, le lecteur de disquette, par exemple, est accessible par le chemin /device/floppy0. Comme nous venons de le voir, le gestionnaire des objets organise de façon hiérarchique les objets. Il dispose pour cela d'objets répertoires. Prenons un exemple d'accès à un fichier sur la disquette : /device/floppy0/projet/NT.doc. La recherche de ce document va se faire en deux étapes :

e) Recherche de l'objet concerné ici le lecteur de disquette (Floppy0),

f) Recherche du fichier par une méthode de l'objet lui-même.

Dans notre cas, c'est un objet du type entrée/sortie qui a une méthode de parcours du disque pour trouver le fichier désiré.

2.4 La sécurité grâce aux objets

L'utilisation d'objets simplifie également grandement les opérations nécessaires à la sécurité. En effet, l'accès à un objet se faisant à travers une de ses procédures liées, il est facile de savoir qui essaie d'effectuer des ouvertures sur cet objet. Par exemple, lorsqu'un processus essaie d'ouvrir un fichier, il appelle la procédure open de l'objet fichier. Rien n'empêche l'objet de vérifier la validité de l'ouverture et/ou d'en garder une trace. De cette manière un audit de l'objet peut être conservé. De plus, le champ quota charges impute à l'utilisateur une limite d'utilisation, ce qui permet à l'administrateur système de répartir les ressources entre les différents utilisateurs.

2.5 Conclusion

L'utilisation des objets permet une gestion homogène et cohérente des ressources systèmes. Les sous-systèmes utilisent les objets de l'exécutive pour fournir les ressources nécessaires à leurs clients. Par exemple, le sous-système Win32 offre à ses clients un service de synchronisation. Ce service est construit sur les objets mutants.

3. Conclusion

Ce projet nous a permis d'appréhender le fonctionnement d'un système d'exploitation récent et novateur qu'est Windows NT. Une bonne culture Unix ou Chorus permet de comprendre rapidement les concepts du noyau. Le micro-noyau de Windows NT étant très fortement inspiré du micro-noyau Mach, nous n'avons trouvé que très peu d'idées réellement novatrices.

D'un point de vue programmation, Microsoft a concentré ses efforts sur l'uniformisation de l'API Win32 pour qu'elle fournisse une boîte à outils complète à l'utilisateur. Beaucoup de fonctions sont d'un très haut niveau masquant ainsi le fonctionnement réel au programmeur. Il nous a donc été très difficile de comprendre les mécanismes internes exacts mis en jeu dans nos applications, le sous-système Win32 masquant toutes les requêtes vers le noyau. Cette difficulté s'est accrue lorsque nous avons voulu créer un thread à distance, service rendu partiellement par l'API Win32.

4. Bibliographie

[CUSTER] Inside Windows NT
Helen CUSTER
Microsoft Press - 1993

[MEYER] Object-oriented Software Construction
Bertrand MEYER
Prentice-Hall International - 1988

[RICHTER] Développer sous Windows 95 et Windows NT 4.0
Jeffrey RICHTER
Microsoft Press - Février 1997

[SCHWAAB] Les systèmes d'exploitation - Concepts et Réalisations
François SCHWAAB et Brigitte WROBEL-DAUTCOURT
Juillet 1993

Index