si l'on veut que la recherche soit rapide. Ainsi, cette table prendra entre 600 Mo et 800 Mo de mémoire principale en permanence, suivant que le système est optimisé pour l'espace ou pour le temps. Ce qui n'est pas très pratique. De toute évidence, le système FAT n'est pas dimensionné pour des gros disques.

Les i-nodes

Pour mémoriser quel bloc appartient à quel fichier, une dernière méthode consiste à associer à chaque fichier une structure de données appelée **nœud d'index** ou **i-node** (*index-node*), laquelle inclut les attributs et les adresses disque des blocs du fichier. Un simple exemple en est montré à la figure 4.13. En fonction de l'i-node, il est ainsi possible de trouver tous les blocs du fichier. Le grand avantage de cette conception par rapport aux fichiers chaînés utilisant une table en mémoire est que l'i-node a uniquement besoin d'être en mémoire quand le fichier correspondant est ouvert. Si chaque i-node occupe n octets, et qu'un maximum de k fichiers puissent être ouverts en même temps, la mémoire totale occupée pour le tableau contenant les i-nodes des fichiers ouverts est seulement de kn octets. Seul cet espace aura besoin d'être réservé à l'avance.

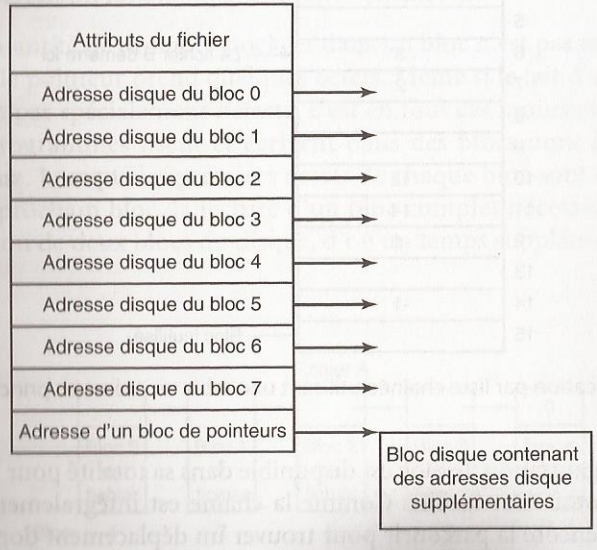

Figure 4.13 • Exemple d'i-node.

Ce tableau est habituellement bien plus petit que l'espace occupé par la table des fichiers de la section précédente. La raison en est simple. La taille de la table de maintien de la liste chaînée de tous les blocs du disque est proportionnelle à celle du disque. Si le disque possède n blocs, la table a besoin de n entrées, et elle croit linéairement à mesure que la taille du disque augmente. *A contrario*, le concept de l'i-node requiert un tableau en mémoire dont la taille est proportionnelle au nombre maximum de fichiers qui peuvent être ouverts simultanément. Que le disque soit de 1, 10 ou 100 Go n'importe pas.

Les i-nodes posent cependant un problème : si chacun d'eux a la place pour un nombre donné d'adresses disque, que se passe-t-il lorsqu'un fichier croit au-delà de cette limite ? Une solution est de réserver la dernière adresse disque, non pour un bloc de données du fichier mais pour l'adresse d'un bloc comprenant plusieurs adresses disque supplémentaires, comme illustré à la figure 4.13. On peut même avoir deux de ces blocs ou plus contenant des adresses disque ou encore des blocs pointant vers d'autres blocs pleins d'adresses disque. Nous reviendrons sur les i-nodes lorsque nous étudierons le système UNIX.

4.3.3 Mise en œuvre des répertoires

Pour lire un fichier, il faut qu'il soit ouvert. Quand un fichier est ouvert, le système d'exploitation se sert du chemin d'accès précisé par l'utilisateur afin de rechercher l'entrée du répertoire. L'entrée du répertoire fournit les informations nécessaires pour trouver les blocs de disque. Suivant les systèmes, ces informations peuvent être l'adresse disque du fichier entier (allocation contiguë), le numéro du premier bloc (cas de la liste chaînée) ou le numéro de l'i-node. Dans tous les cas, la fonction principale du répertoire est de faire correspondre le nom ASCII du fichier à une information nécessaire pour trouver les données.

Un aspect lié de près au point précédent est celui de l'emplacement des attributs. Chaque système de fichiers gère des attributs, tels que le propriétaire de chaque fichier ou sa date de création, et cela doit être stocké quelque part. Une solution évidente consiste à les stocker directement dans l'entrée du répertoire. De nombreux systèmes fonctionnent de la sorte. Cette option est illustrée à la figure 4.14(a). Dans cette conception simple, un répertoire consiste en une liste d'entrées, dont la taille est fixe, qui correspondent chacune à un fichier. Chaque entrée contient le nom du fichier (d'une longueur donnée), une structure des attributs du fichier et une ou plusieurs adresses disque (jusqu'à une valeur maximale) précisant les blocs de disque.

Pour les systèmes qui fonctionnent avec les i-nodes, une autre possibilité de placer les attributs est de le faire dans les i-nodes plutôt que dans les entrées de répertoire. Dans ce cas, l'entrée du répertoire peut être plus petite et être juste constituée d'un nom de fichier et d'un numéro d'i-node. Cette approche est illustrée à la figure 4.14(b). Comme nous le verrons plus loin, cette méthode a certains avantages sur celle qui place les attributs dans l'entrée du répertoire. Les deux approches de la figure 4.14 correspondent respectivement à Windows et UNIX, comme nous le verrons plus tard.

Jusqu'ici, nous avons admis que les fichiers avaient des noms courts de taille fixe. Dans MS-DOS, les noms de fichiers sont composés de 1 à 8 caractères et d'une extension facultative de 1 à 3 caractères. Dans la version 7 d'UNIX, les noms de fichiers sont constitués de 1 à 14 caractères, incluant n'importe quelle extension

Comme l'i-node de la figure 4.13, les i-nodes UNIX contiennent certains attributs : la taille du fichier, trois informations temporelles (date et heure de la création, du dernier accès et de la dernière modification), le nom du propriétaire, le nom du groupe, les informations de protection et un compteur indiquant le nombre d'entrées de répertoire pointant sur lui. Le dernier champ est nécessaire à l'utilisation des liens. À chaque fois qu'un nouveau lien est créé sur un i-node, le compteur de ce dernier est incrémenté de 1, et quand un lien est supprimé, le compteur de l'i-node est décrémenté de 1. Quand sa valeur est nulle, l'i-node est récupéré et les blocs de disque sont placés dans la liste des blocs libres.

La trace des blocs de disque est conservée suivant le principe de la figure 4.13 ; ce principe est toutefois adapté de façon à être applicable à la gestion de très grands fichiers. Les 10 premières adresses des blocs sont enregistrées dans l'i-node lui-même : ainsi, pour les petits fichiers, toutes les informations nécessaires se trouvent dans l'i-node, lequel est chargé dans la mémoire quand le fichier est ouvert. Pour de plus grands fichiers, une des adresses de l'i-node correspond à l'adresse d'un bloc de disque appelé **bloc de simple indirection** (*single indirect block*). Ce bloc contient non pas des données mais des adresses de blocs supplémentaires. Si cela n'est pas suffisant, une autre adresse de l'i-node, appelée **bloc de double indirection** (*double indirect block*), contient l'adresse d'un bloc qui contient une liste de blocs de simple indirection. Chacun de ces blocs de simple indirection pointe sur quelques centaines de blocs de données. Si cela ne suffit toujours pas, on peut avoir recours à un **bloc de triple indirection** (*triple indirect block*). Le schéma complet est donné à la figure 4.34.

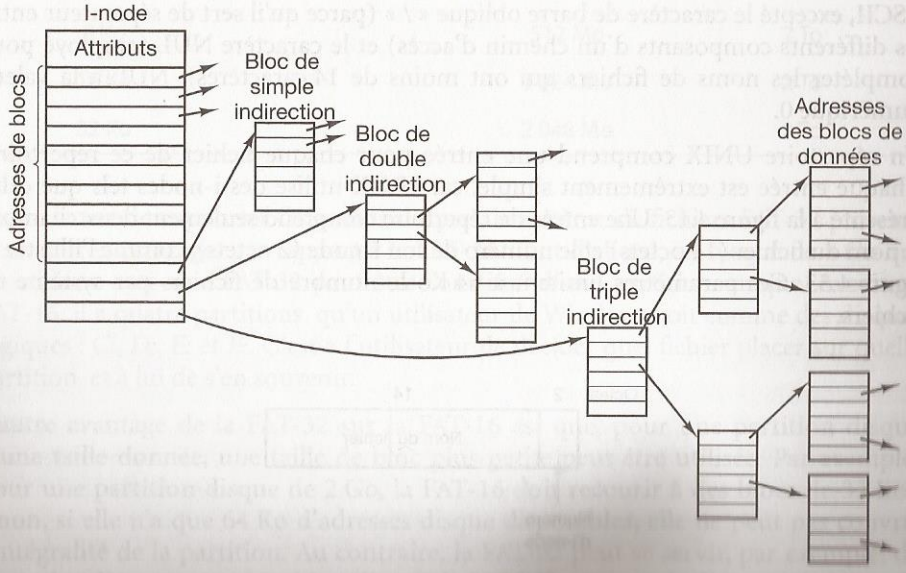

Figure 4.34 • Un i-node UNIX.

ructure du système de fichiers

haque volume NTFS (c'est-à-dire chaque partition) contient des fichiers, des rép
ires, des bitmaps et d'autres structures de données. Chaque volume est organ
mme une séquence de blocs (ou de *clusters*, pour employer la terminologie Mic
ft), avec une taille de bloc fixe pour chaque volume, la valeur pouvant aller d
2 octets à 64 Ko en fonction de la taille du volume. La plupart des disques NT
t une taille de bloc de 4 Ko, ce qui est un bon compromis entre des grands blo
vorisant l'efficacité des transferts et des blocs plus petits, qui occasionnent moins
gmentation interne. Les blocs sont repérés par leur offset à partir du début d
lume, en utilisant des adresses sur 64 bits.

structure de données principale de chaque volume est la **MFT** (*Master File Tab*
le des fichiers maître). C'est une suite d'enregistrements de 1 Ko. Chaque enregi
ment décrit un fichier ou un répertoire. Il contient les attributs du fichier, comme
n nom et les dates d'accès, et la liste des adresses disque de ses blocs. Si un fichier est
aiment très long, on peut avoir besoin de plusieurs enregistrements pour le gérer
r complètement. Dans ce cas, le premier (appelé **enregistrement de base**) pointe
r les suivants. Ce système date de l'époque CP/M, où chaque entrée de répertoire
ait appelée *extent* (extension). Une bitmap permet de repérer les enregistrements
res.

MFT est elle-même un fichier et en tant que telle peut être placée n'importe où
ns le volume, s'affranchissant ainsi des problèmes de secteurs défectueux dans la
mière piste. En outre, le fichier peut croître jusqu'à 2^{48} enregistrements.

MFT est représentée à la figure 11.41. Chaque enregistrement consiste en une
quence de paires *en-tête d'attribut-valeur*. Chaque attribut débute par un en-tête
crivant l'attribut et la longueur de sa valeur. Certains attributs sont de taille variable
, comme le nom du fichier et les données. Si la valeur est suffisamment courte
ur tenir dans l'enregistrement MFT, elle y est placée. On appelle cela un fichier
médiat. Si elle est trop longue, elle est mise ailleurs et on place un pointeur dans
ns l'enregistrement MFT. Cela rend NTFS très efficace pour les petits champs qui
peuvent tenir dans l'enregistrement MFT lui-même.

16 premiers enregistrements MFT sont réservés à des fichiers de métadonnées
FS, comme indiqué à la figure 11.41. Chacun de ces enregistrements décrit un
ier ordinaire, avec ses attributs et ses blocs de données, comme n'importe quel
re fichier. Chacun de ces fichiers possède un nom qui commence par un « $ » pour
iquer qu'il s'agit d'un métafichier.

premier enregistrement décrit le fichier MFT lui-même. En particulier, il indique
nplacement des blocs du fichier afin que le système puisse trouver le fichier
T. Windows a donc clairement besoin d'un moyen de localiser le premier bloc
fichier MFT pour trouver le reste des informations du système de fichiers.
ur localiser le premier bloc du fichier MFT, il examine le bloc de démarrage où
adresse est installée lorsque le volume est formaté avec le système de fichiers.

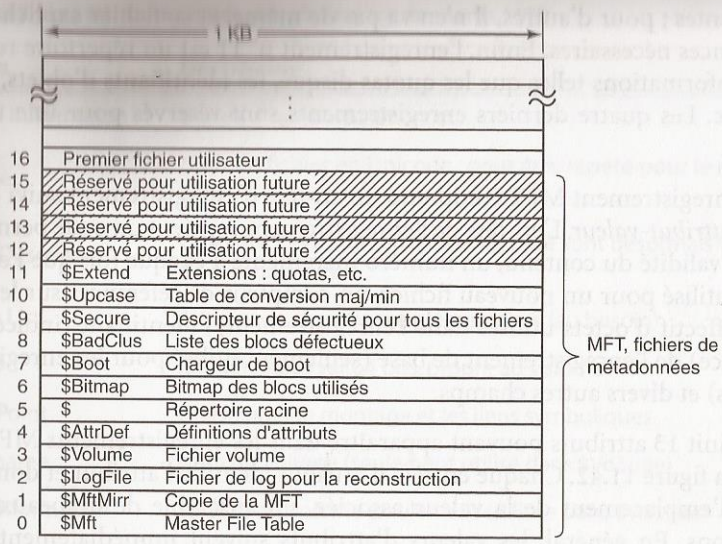

Figure 11.41 • La table de fichiers maître NTFS.

L'enregistrement n° 1 est une copie du début du fichier MFT. Cette information est si précieuse que cette copie est nécessaire en cas de corruption de l'un des premiers blocs de la MFT.

L'enregistrement n° 2 est le fichier de log. Quand des modifications de structure sont apportées au système de fichiers (l'ajout d'un nouveau répertoire ou la suppression d'un répertoire existant, par exemple), elles sont enregistrées avant d'être exécutées, ce qui augmente les chances de récupération correcte en cas de problème pendant l'opération, comme lors d'un crash système. Les modifications d'attributs de fichier sont également journalisées à cet endroit. En fait, les seules modifications non journalisées à cet endroit sont les changements apportés aux données utilisateur. L'enregistrement n° 3 renferme des informations concernant le volume, comme sa taille, son étiquette et sa version.

Comme indiqué précédemment, chaque enregistrement MFT contient une séquence de paires *en-tête d'attribut-valeur*. Le fichier `$AttrDef` contient la définition des attributs présents dans les enregistrements MFT. Les informations sur ce fichier sont regroupées dans l'enregistrement n° 4. On trouve ensuite, dans l'enregistrement n° 5, le répertoire racine, qui est lui-même un fichier et peut grandir sans limitation *a priori*.

L'espace libre dans le volume est repéré à l'aide d'une bitmap, qui est lui-même un fichier dont les attributs et adresses disque sont donnés dans l'enregistrement n° 6. L'enregistrement suivant pointe sur le fichier de chargement du bootstrap. L'enregistrement n° 8 contient la liste de tous les blocs défectueux afin d'en empêcher l'utilisation. L'enregistrement n° 9 renferme les informations de sécurité. L'enregistrement n° 10 décrit les règles d'équivalence de casse. Pour l'alphabet latin (A-Z), ces règles

Le nom du flux, le cas échéant, est dans l'en-tête de cet attribut. Après l'en-tête se trouve une liste d'adresses disque décrivant les blocs que contient le flux ou, pour les petits flux de quelques centaines d'octets (qui sont nombreux), le flux lui-même. Dans ce dernier cas, on parle de **fichier immédiat**.

Il est clair que, la plupart du temps, les données ne sont pas stockées dans l'enregistrement MFT : l'attribut est le plus souvent non résident. Voyons comment NTFS traite les références aux attributs non résidents, en particulier les données.

Allocation de stockage

Pour mémoriser les blocs de disque, on s'appuie sur le fait qu'ils sont généralement alloués en séries consécutives, quand cela est possible, pour des raisons d'efficacité. Ainsi, si le premier bloc d'un flux est placé dans le bloc 20 sur le disque, le système fera tout pour placer les blocs logiques suivants sur les blocs physiques 21 et suivants. Une façon de le faire consiste à allouer plusieurs blocs à la fois lorsque c'est possible.

Les blocs d'un flux sont décrits par une suite d'enregistrements, chaque enregistrement référençant une suite de blocs logiques contigus. Si le flux est sans trou, il n'y aura qu'un enregistrement de ce type. C'est le cas des flux écrits en une fois, du début à la fin. Pour un flux comprenant un trou (admettons, par exemple, que seuls les blocs 0 à 49 et 60 à 79 soient définis), il faudra deux enregistrements. On peut créer un tel flux en écrivant d'abord les 50 premiers blocs, puis en se déplaçant en avant jusqu'au bloc 60 avant d'écrire le reste du fichier. Quand le trou est lu, tous les octets manquants sont considérés comme des zéros. Les fichiers avec des trous sont appelés des **fichiers creux**.

Chaque enregistrement débute par un en-tête donnant le décalage du premier bloc du flux. Juste après, on trouve le décalage du premier bloc non adressé par l'enregistrement. Ainsi, dans le cas précédent, le premier enregistrement contiendrait un en-tête à (0,50) et fournirait l'adresse disque pour ces 50 blocs. Le second enregistrement contiendrait un en-tête à (60,80) et fournirait l'adresse disque pour ces 20 blocs supplémentaires.

Chaque en-tête d'enregistrement est suivi d'une ou plusieurs paires donnant chacune une adresse disque et une longueur. L'adresse disque correspond au décalage du bloc de disque par rapport au début de la partition ; la longueur correspond au nombre de blocs concernés. Il y a autant de paires que nécessaire. La figure 11.43 illustre ce type de schéma pour un flux de neuf blocs répartis en trois paquets.

À cette figure, on trouve un enregistrement MFT pour un flux court de neuf blocs (en-tête 0-8). Il consiste en trois séries de blocs de disque consécutifs. Le premier morceau concerne les blocs 20 à 23, le second les blocs 64 et 65 et le troisième les blocs 80 à 82. Chaque morceau est enregistré dans l'enregistrement MFT sous la forme d'une paire *adresse disque-nombre de blocs*. Selon la qualité du travail de l'allocateur de blocs de disque, un fichier de n blocs peut nécessiter de 1 à n morceaux.

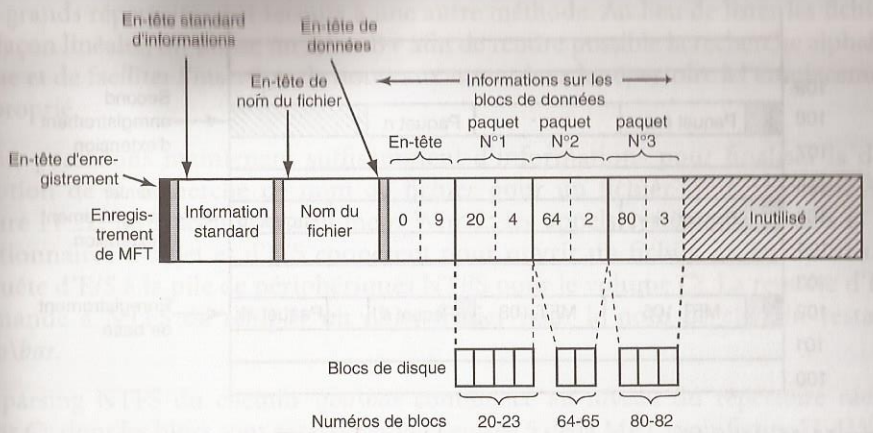

Figure 11.43 • Un enregistrement MFT pour un flux de neufs blocs et trois paquets.

Il convient de faire ici quelques commentaires. Tout d'abord, il n'y a aucune limite à la taille que des flux ainsi représentés peuvent atteindre. Sans compression d'adresse, chaque paire nécessite 2 nombres de 64 bits, donc 16 octets. Cela dit, une paire peut référencer un million de blocs de disque consécutifs, voire plus. Ainsi, un flux de 20 Mo divisé en 20 segments de 1 Mo chacun tient aisément dans un seul enregistrement MFT, alors que ce n'est pas le cas d'un flux de 60 Ko réparti sur 60 blocs isolés.

Ensuite, bien que la méthode de base de représentation de chaque paire prenne 16 octets, il est possible de réduire cette taille par compression. De nombreuses adresses disque ont de nombreux bits de poids fort à 0. Ils peuvent être omis. L'en-tête de données indique combien sont omis – en fait, combien d'octets sont effectivement utilisés pour chaque adresse. D'autres méthodes de compression peuvent également être utilisées. Dans la pratique, les paires se contentent souvent de 4 octets.

Notre premier exemple était très simple, toute l'information du fichier tenait dans un enregistrement MFT. Que se passe-t-il si le fichier est si grand ou si fragmenté que l'information ne tient pas dans un enregistrement unique ? La réponse est simple : on a recours à deux enregistrements, ou plus. La figure 11.44 montre un fichier dont l'enregistrement de base est placé dans l'enregistrement MFT n? 102. Il est formé de trop nombreux morceaux pour tenir dans un enregistrement MFT, il doit donc calculer combien d'enregistrements d'extension seront nécessaires (ici, deux) et place leur indice dans l'enregistrement de base. Le reste de l'enregistrement est utilisé par les k premiers morceaux de données.

Remarquons que la figure 11.44 contient des redondances. En principe, il ne devrait pas être nécessaire de préciser la fin d'une série contiguë de blocs, puisque cette information peut être déduite des paires. La raison de cette précision apparemment excessive a à voir avec l'optimisation de la recherche : pour chercher un bloc à un offset donné, il suffit d'examiner les en-têtes d'enregistrement et pas les paires.