FAT : Aperçu général du format sur disque

Le système de fichiers FAT (File Allocation Table) date de la fin des années 1970 – début des années 1980 et était le système de fichier supporté par le système d'exploitation Microsoft® MS-DOS®. Il a été développé originellement comme un système de fichiers simple utilisable pour des disquettes de moins de 500Ko. Au fil du temps, il a été amélioré pour supporter des médias plus grands. Actuellement, il y a trois types de système de fichiers FAT : FAT12, FAT16 et FAT32. La différence de base entre ces sous types de FAT, et la raison des noms, est la taille, en bits, des entrées dans la structure FAT sur le disque. Il y a 12 bits dans une entrée FAT12, 16 dans une entrée FAT16 et 32 dans une entrée FAT32.

Sommaire

Disclaimer

IMPORTANT—READ CAREFULLY: This Microsoft Agreement (“Agreement”) is a legal agreement between you (either an individual or a single entity) and Microsoft Corporation (“Microsoft”) for this version of the Microsoft specification identified above (“Specification”).
BY DOWNLOADING, COPYING OR OTHERWISE USING THE SPECIFICATION, YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT DOWNLOAD, COPY, OR USE THE SPECIFICATION.

The Specification is owned by Microsoft or its suppliers and is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties.
1. **LIMITED COVENANT NOT TO SUE**.

(a) Provided that you comply with all terms and conditions of this Agreement and subject to the limitations in Sections 1(b) – (e) below, Microsoft grants to you the following non-exclusive, worldwide, royalty-free, nontransferable,
non-sublicenseable, reciprocal limited covenant not to sue:
(i) under any copyrights owned or licensable by Microsoft without payment of consideration to unaffiliated third parties, to reproduce the Specification solely for the purposes of creating portions of products which comply with the Specification in unmodified form; and
(ii) under its Necessary Claims solely to make, have made, use, import, and directly and indirectly, offer to sell, sell and otherwise distribute and dispose of portions of products which comply with the Specification in unmodified form.
For purposes of the foregoing, the Specification is **“unmodified”** if there are no changes, additions or extensions to the Specification, and **“Necessary Claims”** means claims of a patent or patent application which are (1) owned or licenseable by Microsoft without payment of consideration to an unaffiliated third party; and (2) have an effective filing date on or before December 31, 2010, that must be infringed in order to make a portion(s) of a product that complies with the Specification. Necessary Claims does not include claims relating to semiconductor manufacturing technology or microprocessor circuits or claims not required to be infringed in complying with the Specification (even if in the same patent as Necessary Claims).

(b) The foregoing covenant not to sue shall not extend to any part or function of a product which (i) is not required to comply with the Specification in unmodified form, or (ii) to which there was a commercially reasonable alternative to infringing a Necessary Claim.

(c) The covenant not to sue described above shall be unavailable to you and shall terminate immediately if you or any of your Affiliates (collectively “Covenantee Party”) “Initiates” any action for patent infringement against: (x) Microsoft or any of its Affiliates (collectively “Granting Party”), (y) any customers or distributors of the Granting Party, or other recipients of a covenant not to sue with respect to the Specification from the Granting Party (“Covenantees”); or (z) any customers or distributors of Covenantees (all parties identified in (y) and (z) collectively referred to as “Customers”), which action is based on a conformant implementation of the Specification. As used herein, “Affiliate” means any entity which directly or indirectly controls, is controlled by, or is under common control with a party; and control shall mean the power, whether direct or indirect, to direct or cause the direct of the management or policies of any entity whether through the ownership of voting securities, by contract or otherwise. “Initiates” means that a Covenantee Party is the first (as between the Granting Party and the Covenantee Party) to file or institute any legal or administrative claim or action for patent infringement against the Granting Party or any of the Customers. “Initiates” includes any situation in which a Covenantee Party files or initiates a legal or administrative claim or action for patent infringement solely as a counterclaim or equivalent in response to a Granting Party first filing or instituting a legal or administrative patent infringement claim against such Covenantee Party.

(d) The covenant not to sue described above shall not extend to your use of any portion of the Specification for any purpose other than (a) to create portions of an operating system (i) only as necessary to adapt such operating system so that it can directly interact with a firmware implementation of the Extensible Firmware Initiative Specification v. 1.0 (“EFI Specification”); (ii) only as necessary to emulate an implementation of the EFI Specification; and (b) to create firmware, applications, utilities and/or drivers that will be used and/or licensed for only the following purposes: (i) to install, repair and maintain hardware, firmware and portions of operating system software which are utilized in the boot process; (ii) to provide to an operating system runtime services that are specified in the EFI Specification; (iii) to diagnose and correct failures in the hardware, firmware or operating system software; (iv) to query for identification of a computer system (whether by serial numbers, asset tags, user or otherwise); (v) to perform inventory of a computer system; and (vi) to manufacture, install and setup any

hardware, firmware or operating system software.
(e) Microsoft reserves all other rights it may have in the Specification and any intellectual property therein. The furnishing of this document does not give you any covenant not to sue with respect to any other Microsoft patents, trademarks, copyrights or other intellectual property rights; or any license with respect to any Microsoft intellectual property rights.

2. **ADDITIONAL LIMITATIONS AND OBLIGATIONS**.

(a) The foregoing covenant not to sue is applicable only to the version of the Specification which you are about to download. It does not apply to any additional versions of or extensions to the Specification.

(b) Without prejudice to any other rights, Microsoft may terminate this Agreement if you fail to comply with the terms and conditions of this Agreement. In such event you must destroy all copies of the Specification.

3. **INTELLECTUAL PROPERTY RIGHTS**. All ownership, title and intellectual property rights in and to the Specification are owned by Microsoft or its suppliers.

4.**U.S. GOVERNMENT RIGHTS**. Any Specification provided to the U.S. Government pursuant to solicitations issued on or after December 1, 1995 is provided with the commercial rights and restrictions described elsewhere herein. Any Specification provided to the U.S. Government pursuant to solicitations issued prior to December 1, 1995 is provided with RESTRICTED RIGHTS as provided for in FAR, 48 CFR 52.227-14 (JUNE 1987) or DFAR, 48 CFR 252.227-7013 (OCT 1988), as applicable.

5.**EXPORT RESTRICTIONS**. Export of the Specification, any part thereof, or any process or service that is the direct product of the Specification (the foregoing collectively referred to as the "Restricted Components") from the United States is regulated by the Export Administration Regulations (EAR, 15 CFR 730-744) of the U.S. Commerce Department, Bureau of Export Administration ("BXA"). You agree to comply with the EAR in the export or re-export of the Restricted Components (i) to any country to which the U.S. has embargoed or restricted the export of goods or services, which currently include, but are not necessarily limited to Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria and the Federal Republic of Yugoslavia (including Serbia, but not Montenegro), or to any national of any such country, wherever located, who intends to transmit or transport the Restricted Components back to such country; (ii) to any person or entity who you know or have reason to know will utilize the Restricted Components in the design, development or production of nuclear, chemical or biological weapons; or (iii) to any person or entity who has been prohibited from participating in U.S. export transactions by any federal agency of the U.S. government. You warrant and represent that neither the BXA nor any other U.S. federal agency has suspended, revoked or denied your export privileges. For additional information see http://www.microsoft.com/exporting.

6. **DISCLAIMER OF WARRANTIES. To the maximum extent permitted by applicable law, Microsoft and its suppliers provide the Specification (and all intellectual property therein) and any (if any) support services related to the Specification ("Support Services")** ***AS IS AND WITH ALL FAULTS*****, and hereby disclaim all warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties or conditions of merchantability, of fitness for a particular purpose, of lack of viruses, of accuracy or completeness of responses, of results, and of lack of negligence or lack of workmanlike effort, all with regard to the Specification, any intellectual property therein and the provision of or failure to provide Support Services. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT, WITH REGARD TO THE SPECIFICATION AND ANY INTELLECTUAL PROPERTY THEREIN. THE ENTIRE RISK AS TO THE QUALITY OF OR ARISING OUT OF USE OR PERFORMANCE OF THE SPECIFICATION, ANY INTELLECTUAL PROPERTY THEREIN, AND SUPPORT SERVICES, IF ANY, REMAINS WITH YOU.**
7. **EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL MICROSOFT OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SPECIFICATION, ANY INTELLECTUAL PROPERTY THEREIN, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS AGREEMENT, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, BREACH OF CONTRACT OR BREACH OF WARRANTY OF MICROSOFT OR ANY SUPPLIER, AND EVEN IF MICROSOFT OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.**
8. **LIMITATION OF LIABILITY AND REMEDIES. Notwithstanding any damages that you might incur for**

any reason whatsoever (including, without limitation, all damages referenced above and all direct or general damages), the entire liability of Microsoft and any of its suppliers under any provision of this Agreement and your exclusive remedy for all of the foregoing shall be limited to the greater of the amount actually paid by you for the Specification or U.S.$5.00. The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails its essential purpose.

9. **APPLICABLE LAW**. If you acquired this Specification in the United States, this Agreement is governed by the laws of the State of Washington. If you acquired this Specification in Canada, unless expressly prohibited by local law, this Agreement is governed by the laws in force in the Province of Ontario, Canada; and, in respect of any dispute which may arise hereunder, you consent to the jurisdiction of the federal and provincial courts sitting in Toronto, Ontario. If this Specification was acquired outside the United States, then local law may apply.

10.**QUESTIONS**. Should you have any questions concerning this Agreement, or if you desire to contact Microsoft for any reason, please contact the Microsoft subsidiary serving your country, or write: Microsoft Sales Information Center/One Microsoft Way/Redmond, WA 98052-6399.

11.**ENTIRE AGREEMENT. This Agreement is the entire agreement between you and Microsoft relating to the Specification and the Support Services (if any) and they supersede all prior or contemporaneous oral or written communications, proposals and representations with respect to the Specification or any other subject matter covered by this Agreement. To the extent the terms of any Microsoft policies or programs for Support Services conflict with the terms of this Agreement, the terms of this Agreement shall control.**

Si vous avez acquis votre produit Microsoft au CANADA, la garantie limitée suivante vous concerne :

RENONCIATION AUX GARANTIES. Dans toute la mesure permise par la législation en vigueur, Microsoft et ses fournisseurs fournissent la Specification (et à toute propriété intellectuelle dans celle-ci) et tous (selon le cas) les services d'assistance liés à la Specification ("Services d'assistance") TELS QUELS ET AVEC TOUS LEURS DÉFAUTS, et par les présentes excluent toute garantie ou condition, expresse ou implicite, légale ou conventionnelle, écrite ou verbale, y compris, mais sans limitation, toute (selon le cas) garantie ou condition implicite ou légale de qualité marchande, de conformité à un usage particulier, d'absence de virus, d'exactitude et d'intégralité des réponses, de résultats, d'efforts techniques et professionnels et d'absence de négligence, le tout relativement à la Specification, à toute propriété intellectuelle dans celle-ci et à la prestation ou à la non-prestation des Services d'assistance. DE PLUS, IL N'Y A AUCUNE GARANTIE ET CONDITION DE TITRE, DE JOUISSANCE PAISIBLE, DE POSSESSION PAISIBLE, DE SIMILARITÉ À LA DESCRIPTION ET D'ABSENCE DE CONTREFAÇON RELATIVEMENT À LA SPÉCIFICATION ET À TOUTE PROPRIÉTÉ INTELLECTUELLE DANS CELLE-CI. VOUS SUPPORTEZ TOUS LES RISQUES DÉCOULANT DE L'UTILISATION ET DE LA PERFORMANCE DE LA SPÉCIFICATION ET DE TOUTE PROPRIÉTÉ INTELLECTUELLE DANS CELLE-CI ET CEUX DÉCOULANT DES SERVICES D'ASSISTANCE (S'IL Y A LIEU).

EXCLUSION DES DOMMAGES INDIRECTS, ACCESSOIRES ET AUTRES. Dans toute la mesure permise par la législation en vigueur, Microsoft et ses fournisseurs ne sont en aucun cas responsables de tout dommage spécial, indirect, accessoire, moral ou exemplaire quel qu'il soit (y compris, mais sans limitation, les dommages entraînés par la perte de bénéfices ou la perte d'information confidentielle ou autre, l'interruption des affaires, les préjudices corporels, la perte de confidentialité, le défaut de remplir toute obligation y compris les obligations de bonne foi et de diligence raisonnable, la négligence et toute autre perte pécuniaire ou autre perte de quelque nature que ce soit) découlant de, ou de toute autre manière lié à, l'utilisation ou l'impossibilité d'utiliser la Spécification, toute propriété intellectuelle dans celle-ci, la prestation ou la non-prestation des Services d'assistance ou autrement en vertu de ou relativement à toute disposition de cette convention, que ce soit en cas de faute, de délit (y compris la négligence), de responsabilité stricte, de manquement à un contrat ou de manquement à une garantie de Microsoft ou de l'un de ses fournisseurs, et ce, même si Microsoft ou l'un de ses fournisseurs a été avisé de la possibilité de tels dommages.

LIMITATION DE RESPONSABILITÉ ET RECOURS. Malgré tout dommage que vous pourriez encourir pour quelque raison que ce soit (y compris, mais sans limitation, tous les dommages mentionnés ci-dessus et tous les dommages directs et généraux), la seule responsabilité de Microsoft et de ses fournisseurs en vertu de toute disposition de cette convention et votre unique recours en regard de tout ce qui précède sont limités au plus élevé des montants suivants: soit (a) le montant que vous avez payé pour la Spécification, soit (b) un montant équivalant à cinq dollars U.S. (5,00 $ U.S.). Les limitations, exclusions et renonciations ci-dessus s'appliquent dans toute la mesure permise par la législation en vigueur, et ce même si leur application a pour effet de priver un recours de son essence.

DROITS LIMITÉS DU GOUVERNEMENT AMÉRICAIN
Tout Produit Logiciel fourni au gouvernement américain conformément à des demandes émises le ou après le 1er décembre 1995 est offert avec les restrictions et droits commerciaux décrits ailleurs dans la présente

convention. Tout Produit Logiciel fourni au gouvernement américain conformément à des demandes émises avant le 1er décembre 1995 est offert avec des DROITS LIMITÉS tels que prévus dans le FAR, 48CFR 52.227-14 (juin 1987) ou dans le FAR, 48CFR 252.227-7013 (octobre 1988), tels qu'applicables. Sauf lorsqu'expressément prohibé par la législation locale, la présente convention est régie par les lois en vigueur dans la province d'Ontario, Canada. Pour tout différend qui pourrait découler des présentes, vous acceptez la compétence des tribunaux fédéraux et provinciaux siégeant à Toronto, Ontario.

Si vous avez des questions concernant cette convention ou si vous désirez communiquer avec Microsoft pour quelque raison que ce soit, veuillez contacter la succursale Microsoft desservant votre pays, ou écrire à: Microsoft Sales Information Center, One Microsoft Way, Redmond, Washington 98052-6399.

Conventions utilisées dans ce document

Les nombres commençant par les caractères ‘0x’ sont exprimés en hexadécimal (base 16).

Tout nombre qui ne commence pas par ‘0x’ sont exprimé en décimal (base 10).

Les fragments de code de ce document sont écrits en langage de programmation C. La syntaxe et la graphie strictes ne sont pas forcément employées.

Il y a plusieurs fragments de code dans ce document qui mélangent librement des données 16 bits et 32 bits. Nous supposons que vous êtes un programmeur qui sait comment saisir de telles opérations afin qu’il n’y ai pas de perte de données due à la troncation de valeurs 16 bits vers des valeurs 32 bits. Prenez également note que tous les types de données sont NON SIGNES. Ne faites pas de calculs FAT avec des entiers signés car ils seraient faux sur certains volumes FAT.

Commentaires généraux (applicables à tout type de FAT)

Tous les systèmes de fichiers FAT on été initialement développés pour une architecture PC IBM. De ce fait, toutes les structures de données sur disque du système de fichiers FAT sont au format ‘little endian’. Si nous voyons une entrée FAT 32 bits stockée sur un disque comme une série de 4 octets, le premier étant octet[0] et le dernier octet[3], voici comment sont organisés les bits numérotés de 00 à 31 (00 étant le bit le moins significatif) :

```
octet[3]        3 3 2 2 2 2 2 2
                1 0 9 8 7 6 5 4
octet[2]        2 2 2 2 1 1 1 1
                3 2 1 0 9 8 7 6
octet[1]        1 1 1 1 1 1 0 0
                5 4 3 2 1 0 9 8
octet[0]        0 0 0 0 0 0 0 0
                7 6 5 4 3 2 1 0
```

C’est important si votre machine est une machine ‘big endian’, car vous aurez à passer de l’un à l’autre lorsque vous envoyez ou lisez des données depuis le disque.

Un volume FAT est composé de quatre zones de base, qui sont placés dans cet ordre sur le volume :

0 – zone réservée
1 – zone FAT
2 – zone du répertoire racine (sauf sur les volumes FAT32)
3 – zones de fichiers et de répertoires

Secteur de boot et BPB

La première structure de données importante sur un volume FAT est appelée BPB (bloc de paramètres du BIOS, BIOS Parameter Block), qui est situé sur le premier secteur du volume dans la zone réservée. Ce secteur est quelquefois appelé secteur de boot ou secteur réservé ou secteur 0, mais le fait important est simplement que c'est le premier secteur du volume.

C'est la première chose sur le système de fichiers FAT qui peut parfois porter à confusion. Sous MS-DOS v1.x, il n'y avait pas de BPB sur le secteur de boot. Dans la première version du système de fichiers FAT, il n'y avait que deux formats différents pour les disquettes 5'¼, le premier simple face et le second double face. Pour déterminer de quel type était la disquette, on regardait le premier octet de la FAT (des 8 bits de poids faible de FAT[0]).

Ce type de détermination a été remplacé avec MS-DOS 2.x en mettant le BPB dans le secteur de boot, et l'ancien type de détermination n'a plus été supporté. Tous les volumes FAT doivent avoir un BPB dans le secteur de boot.

Cela nous conduit au second point portant à confusion sur la détermination du volume FAT : A quoi ressemble exactement le BPB ? Le BPB du secteur de boot défini pour MS-DOS 2.x ne permettait d'avoir que des volumes avec moins de 65536 secteurs (32Mo avec des secteurs de 512 octets). Cette limitation était due au fait que le champ 'nombre total de secteurs' ne faisait que 16 bits, elle a été repoussée avec MS-DOS 3.x, où le BPB a été modifié pour inclure un champ de 32 bits pour le nombre total de secteurs.

Le changement suivant dans la BPB est intervenu avec le système d'exploitation Microsoft Windows 95, avec lequel la FAT32 est apparue. La FAT16 était limitée par la taille maximale de la FAT et la taille maximum d'un cluster à des volumes de 2Go maximum (avec des secteurs de 512 octets). La FAT32 a augmenté l'espace que peut occuper la FAT sur un volume afin que des disques de plus de 2Go puissent avoir une seule partition de définie.

Le BPB de la FAT32 est exactement identique aux BPB de la FAT12 et de la FAT16 ci dessus et inclus le champ du total de secteurs sur 32 bits. Ils commencent à différer à partir de l'octet 36 suivant que le média est de type FAT12/FAT16 ou FAT32 (voir ci dessous pour déterminer le type de FAT). Le point marquant ici est que le BPB dans le secteur de boot d'un volume FAT devrait toujours avoir tous les nouveaux champs, qu'il soit de type FAT12/FAT16 ou FAT32. Cela assure une compatibilité maximale entre les volumes FAT et assure que tous les pilotes FAT comprendront et supporterons le volume correctement car il contient tous les champs actuellement définis.

NOTE : dans la description suivante, tous les camps qui commencent par BPB_ font partie du BPB. Tous les champs dont le nom commence par BS_ font partie du secteur de boot mais pas réellement du BPB. Ce qui suit montre le début du secteur 0 d'un volume FAT qui contient le BPB.

Structure du secteur de boot et du BPB

Nom	Déplacement (octets)	Taille (octets)	Description
BS_jmpBoot	0	3	Instruction de saut vers le code de démarrage. Ce champ a deux formes autorisées : **jmpBoot[0] = 0xEB, jmpBoot[1] = ??, jmpBoot[2] = 0x90** et **jmpBoot[0] = 0xE9, jmpBoot[1] = ??, jmpBoot[2] = 0x ??** 0x ?? indique qu'une valeur 8 bits quelconque est autorisée pour cet octet. Cela forme un branchement inconditionnel Intel de 3 octets qui saute au début du code d'amorçage du système. Typiquement, ce code occupe le reste du secteur 0 du volume suivant le BPB et éventuellement d'autres secteurs. Chacun des formats présentés ci dessus est accepté. jmpBoot[0] = 0xEB est le plus fréquent.
BS_OEMName	3	8	'MSWIN4.1' Il y a plusieurs malentendus à propos de ce champ. C'est seulement une chaîne. Les systèmes Microsoft ne tiennent pas compte de ce champ. Quelques pilotes FAT le font. C'est la raison pour laquelle la chaîne présentée est le paramètre recommandé. Car c'est celui qui risque le moins de présenter des problèmes d'incompatibilité. Si vous voulez mettre quelque chose d'autre ici, c'est votre choix, mais certains pilotes FAT pourraient ne pas reconnaître le volume. Typiquement, cela indique quel système a formaté le disque.
BPB_BytsPerSec	11	2	Nombre d'octets par secteur. Cette valeur peut être une des suivantes : 512, 1024, 2048 ou 4096. Si l'on vise la compatibilité maximale, la valeur 512 doit être utilisée. Il y a beaucoup de code FAT dans le monde pour lequel la valeur 512 est codée en dur et qui ne le vérifient pas. Les systèmes Microsoft supporteront correctement 1024, 2048 et 4096 mais ces valeurs ne sont pas recommandées.
BPB_SecPerClus	13	1	Nombre de secteurs par unité d'allocation. Cette valeur doit être une puissance de 2 supérieure à 0. Les valeurs valides sont 1, 2, 4, 8, 16, 32, 64 et 128. Notez cependant qu'une valeur donnant un nombre d'octets par cluster (BPB_BytsPerSec*BPB_SecPerClus) supérieur à 32K (32*1024) ne devrait pas être utilisée. Il y a un malentendu quant au fait que des valeurs plus grandes sont valides. Les valeurs qui créent une taille de cluster supérieure à 32 Ko ne fonctionnent pas correctement ; n'essayez pas d'en définir une. Quelques versions de systèmes autorisent 64Ko par cluster. Beaucoup d'applications ne fonctionneront pas sur un volume FAT de ce type.
BPB_RsvdSecCnt	14	2	Nombre de secteurs réservés dans la zone réservée du volume commençant au premier secteur du volume. Ce champ ne doit pas être à 0. Pour les volumes FAT12 et FAT16, cette valeur ne devrait pas être autre chose que 1. Pour les volumes FAT32, c'est typiquement 32. Il y a beaucoup de pilotes FAT pour lesquels la valeur 1est codée en dur pour les volumes FAT12 et FAT16 et qui ne le vérifient pas. Les systèmes Microsoft supporteront toute valeur différente de 0.

Nom	Déplacement	Taille	Description
BPB_NumFATs	16	1	Nombre de structures de données FAT sur le volume. Ce champ devrait toujours contenir la valeur 2 pour tout volume FAT quelque soit le type. Cependant, toute valeur supérieure ou égale à 1 est parfaitement valide, beaucoup de logiciel et quelques systèmes d'exploitation ne fonctionneront pas correctement si cette valeur n'est pas 2. Tous les pilotes Microsoft supporteront une valeur autre que 2, mais il est quand même recommandé de ne pas mettre autre chose. La raison de la valeur standard 2 est prévue pour avoir une FAT redondante afin que si un secteur est corrompu sur une des deux FAT, les données ne soient pas perdues grâce à la deuxième FAT. Sur des medias de type autre que disque, comme les carte mémoires FLASH, ou une telle redondance est obsolète, la valeur 1 peut être utilisée afin d'économiser l'espace que prendrait une seconde FAT. Mais quelques pilotes pourraient ne pas reconnaître un tel volume correctement
BPB_RootEntCnt	17	2	Pour les volumes FAT12 et FAT16, ce champ contient le nombre d'entrées de répertoires de 32 octets dans le répertoire racine. Pour les volumes FAT32, ce champ doit être à 0. Pour les volumes FAT12 et FAT16, cette valeur devrait toujours donner un multiple de BPB_BytsPerSec. Pour une compatibilité maximum, les volumes FAT16 devraient utiliser une valeur de 512.
BPB_TotSec16	19	2	Ce champ est l'ancien total de secteur sur 16 bits. Ce compte comprend tous les secteurs des quatre zones du volume. Ce champ peut être à 0, si c'est la cas, BPB_TotSec32 doit être différent de 0. Pour les volumes FAT32, ce champ doit être à 0. Pour les volumes FAT12/FAT16, ce champ contient le total des secteurs et BPB_TotSec32 est 0 si le total tient (s'il est inférieur à 0x10000).
BPB_Media	21	1	0xF8 est la valeur standard pour les médias non amovibles. Pour les autres, 0xF0 est fréquemment utilisé. Les valeurs valides pour ce champ sont 0xF0 et 0xF8 à 0xFF. Le seul point important à noter est que quelque soit la valeur placée ici, elle doit l'être aussi dans le premier octet de poids faible de l'entrée FAT[0]. Cela date de l'ancienne détermination de média de MS-DOS 1.x signalée plus tôt et n'est plus utilisé pour quoi que ce soit.
BPB_FATSz16	22	2	Ce champ est le nombre de secteurs occupé par UNE FAT en FAT12/FAT16. Sur les volumes FAT32, ce champ doit être à 0 et BPB_FATSz32 contient la taille d'une FAT.
BPB_SecPerTrk	24	2	Secteurs par piste pour l'interruption 0x13. Ce champ est pertinent uniquement pour les médias qui ont une géométrie (le volume est divisé en plusieurs pistes avec des têtes et des cylindres) et sont visibles lors de l'interruption 0x13. Ce champ contient la valeur 'secteur par piste' de la géométrie.
BPB_NumHeads	26	2	Nombre de têtes pour l'interruption 0x13. Ce champ est pertinent dans les mêmes cas que BPB_SecPerTrk. Ce champ contient la valeur 'nombre de têtes'. Par exemple, pour une disquette 3'½ 1,44Mo, sa valeur est 2
BPB_HiddSec	28	4	Nombre de secteurs précédant la partition qui contient le volume FAT. Ce champ n'est généralement pertinent que pour les médias visibles par l'interruption 0x13. Ce champ devrait toujours être à 0 pour les médias non partitionnés. La valeur exacte à spécifier est propre au système d'exploitation
BPB_TotSec32	32	4	Ce champ de 32 bits est le nouveau nombre total de secteurs sur le volume. Ce total inclus tous les secteurs des quatre zones du volume. Ce champ peut être à 0, si c'est le cas, BPB_TotSec16 doit être renseigné. Pour les volumes FAT12/FAT16, ce champ contient le nombre total de secteur si BPB_TotSec16 est à 0 (plus grand que 0x10000).

A partir d'ici, les BPB/secteur de boot pour les FAT12/FAT16 et la FAT32 diffèrent. Le premier tableau montre leur structure pour les FAT12/FAT16 en commençant au déplacement 36.

Structure FAT12 et FAT16 à partir du déplacement 36

Nom	Déplacement (octets)	Taille (octets)	Description
BS_DrvNum	36	1	Numéro de lecteur de l'interruption 0x13 (par exemple 0x80). Ce champ supporte l'amorçage MS-DOS et est définit au numéro de lecteur du média donné par l'interruption 0x13. **NOTE :** ce champ est actuellement propre au système.
BS_Reserved1	37	1	Réservé (utilisé par Windows NT). Le code qui formate les volumes FAT devrait toujours mettre ce bit à 0.
BS_BootSig	38	1	Signature de boot étendue (0x29). C'est un octet de signature qui indique que les trois champs suivants sont présents.
BS_VolID	39	4	Numéro de série du volume. Ce champ, avec BS_VolLab permet le suivi de volume sur les médias amovibles. Ces valeurs permettent au pilote FAT de détecter qu'une mauvaise disquette a été insérée. Cet identifiant est habituellement générée en combinant simplement la date et l'heure courante dans une valeur 32 bits.
BS_VolLab	43	11	Nom de volume. Ce champ correspond au nom de volume enregistré dans le répertoire racine. NOTE : les pilotes FAT devraient s'assurer qu'ils mettent à jour ce champ quand le nom de volume est créé ou changé dans le répertoire racine. Le paramètre pour ce champ lorsqu'il n'y a pas de nom de volume est 'NO NAME ' .
BS_FilSysType	54	8	Une des chaînes 'FAT12 ', 'FAT16 ' ou 'FAT16 '. NOTE : beaucoup de personnes pensent que la chaîne présente dans ce champ a quelque chose à voir avec la détermination du type de FAT utilisé (FAT12, FAT16 ou FAT32). Ce n'est pas vrai. Vous noterez d'après son nom que ce champ ne fait pas partie du BPB. Cette chaîne est là pour information et les drivers Microsoft ne l'utilisent pas car elle est souvent renseignée de manière incorrecte ou absente. Voyez la section 'Détermination du type de FAT' de ce document. Cette chaîne devrait être renseignée d'après le type de FAT malgré tout car certains drivers non Microsoft l'utilisent.

Voici la structure de la FAT32 à partir du déplacement 36 du secteur de boot.

Structure FAT32 à partir du déplacement 36

Nom	Déplacement	Taille	Description
BPB_FATSz32	36	4	Ce champ n'est défini que pour les volumes FAT32 et n'existe pas sur les médias FAT12 et FAT16. Ce champ est le nombre de secteurs occupé par UNE FAT sur 32 bits. BPB_FATSz16 doit être à 0.
BPB_ExtFlags	40	2	Ce champ n'est défini que pour les volumes FAT32 et n'existe pas sur les médias FAT12 et FAT16. Bits 0-3 : numéro de la FAT active (à partir de 0). Valide uniquement si le mirroring est désactivé Bits 4-6 : réservés Bits 7 : 0 signifie que la FAT est reproduites à tout instant sur toutes les FATs 1 signifie que seule une FAT est active ; celle référencée par les bits 0-3 Bits 8-15 : réservés
BPB_FSVer	42	2	Ce champ n'est défini que pour les volumes FAT32 et n'existe pas sur les médias FAT12 et FAT16. L'octet de poids fort est le numéro de version majeur, l'octet de poids faible est le numéro de version mineur. C'est le numéro de version du volume FAT32. Cela offre la possibilité de créer de nouvelles versions de FAT dans le futur sans s'inquiéter des anciens pilotes FAT qui monteront le volume. Ce document décrit la version 0.0, si ce champ est différent de 0, les anciennes versions de Windows ne monteront pas le volume. **NOTE :** les utilitaires disque devraient respecter ce champ et ne pas opérer sur une version autre que celle pour laquelle ils ont été conçus. Les pilotes FAT32 doivent vérifier ce champ et refuser de monter des volumes possédant une version différente de celle pour laquelle ils ont été conçus.
BPB_RootClus	44	4	Ce champ n'est défini que pour les volumes FAT32 et n'existe pas sur les médias FAT12 et FAT16. Il contient le numéro de cluster du répertoire racine, habituellement 2 mais pas forcément. **NOTE :** les utilitaires disque qui changent l'emplacement du répertoire racine devraient tout faire pour qu'il soit placé sur le premier cluster valide du média (CAD le cluster 2 à moins qu'il ne soit marqué comme étant invalide). Cela permet aux utilitaires de récupération de retrouver facilement le répertoire racine si ce champ est corrompu.
BPB_FSInfo	48	2	Ce champ n'est défini que pour les volumes FAT32 et n'existe pas sur les médias FAT12 et FAT16. Numéro du secteur contenant la structure FSINFO dans la zone réservée du média. Habituellement 1. NOTE : il y aura une copie de la structure FSINFO dans BackupBoot, mais seule la copie référencée par ce champ sera maintenue à jour (c'est à dire qu'à la fois l'enregistrement de démarrage et celui de sauvegarde pointeront vers le même secteur FSINFO).
BPB_BkBootSec	50	2	Ce champ n'est défini que pour les volumes FAT32 et n'existe pas sur les médias FAT12 et FAT16. S'il est différent de 0, il indique le numéro du secteur de la zone réservée du volume contenant une copie du secteur de boot. Habituellement 6, aucune valeur différente n'est recommandée.
BPB_Reserved	52	12	Ce champ n'est défini que pour les volumes FAT32 et n'existe pas sur les médias FAT12 et FAT16. Réservé pour une future évolution. Le code formatant les volumes FAT32 devraient toujours initialiser les bits de ce champ à 0
BS_DrvNum	64	1	Ce champ a la même fonction que sur les volumes FAT12/FAT16. La seule différence est le déplacement.
BS_Reserved1	65	1	Ce champ a la même fonction que sur les volumes FAT12/FAT16. La seule différence est le déplacement.
BS_BootSig	66	1	Ce champ a la même fonction que sur les volumes FAT12/FAT16. La seule différence est le déplacement.

Nom	Déplacement	Taille	Description
BS_BootSig	66	1	Ce champ a la même fonction que sur les volumes FAT12/FAT16. La seule différence est le déplacement.
BS_VolId	67	4	Ce champ a la même fonction que sur les volumes FAT12/FAT16. La seule différence est le déplacement.
BS_VolLab	71	11	Ce champ a la même fonction que sur les volumes FAT12/FAT16. La seule différence est le déplacement.
BS_FilSysType	82	8	Toujours renseigné avec la chaîne 'FAT32 '. Veuillez consulter la note sur ce champ dans la section sur la FAT12/FAT16. Ce champ n'a rien à voir avec la détermination du type de FAT

Il y a une autre chose importante à noter à propos du secteur 0 d'un volume FAT. Si nous considérons le contenu du secteur comme un tableau d'octets (dont le premier est secteur[0]), les deux choses suivantes doivent être vérifiées : secteur[510] = 0x55 et secteur[511] = 0xAA.

NOTE : Beaucoup de documents sur la FAT indiquent par erreur que cette signature 0x55AA occupe les 'deux derniers octets du secteur de boot'. Cette affirmation est correcte si et seulement si BPB_BytsPerSec vaut 512. S'il est plus grand que 512, le déplacement de cette signature ne change pas (bien qu'il soit correct de placer cette signature dans les deux derniers octets du secteur de démarrage).

Vérifiez toujours la valeur de BPB_TotSec16/32. Supposons que nous ayons une partition qui ait une taille de DskSz. Si le champ BPB_TotSec (soit BPB_TotSec16, soit BPB_TotSec32) est *inférieur ou égal* à DskSz, il n'y a rien de mauvais avec le volume FAT. En fait, il n'est pas du tout inhabituel d'avoir une valeur BPB_TotSec16/32 qui soit légèrement inférieur à DskSz. Il est également parfaitement correct d'avoir une valeur considérablement inférieure à DskSz.

Tout ce que cela signifie est que de l'espace est gaspillé. Cela ne signifie pas en soit que le volume FAT est endommagé d'une quelconque manière. Par contre, si BPB_TotSec16/32 est *plus grand* que DskSz, le volume est sérieusement endommagé ou déformé puisqu'il s'étend au delà de la fin physique du média et qu'il écrase les données qui le suivent. Traiter normalement un volume pour lequel la valeur BPB_TotSec16/32 est 'trop grande' pour la partition peut conduire à des pertes de données catastrophiques.

Structure de données FAT

La prochaine structure de données importante est la FAT elle-même. Cette structure définit une liste chaînée des clusters d'un fichier. Notez qu'un répertoire FAT n'est rien d'autre qu'un fichier normal avec un attribut spécial indiquant que c'est un répertoire. La seule autre chose spéciale à propos des répertoires est que le contenu de ce 'fichier' est une série d'entrée de répertoire FAT de 32 octets (voir le détail ci dessous). Sous tous les autres aspects, un répertoire est identique à un fichier. La FAT est une cartographie du volume cluster par cluster. Le premier cluster de données est le numéro 2.

Le premier secteur du cluster 2 (la zone de données du disque) est calculé en utilisant les champs BPB du volume comme suit. D'abord, nous déterminons le nombre de secteurs occupés par le répertoire racine :

```
RootDirSectors = ((BPB_RootEntCnt * 32) + (BPB_BytsPerSec – 1)) / BPB_BytsPerSec;
```

Notez que sur les volumes FAT32 la valeur BPB_RootEntCnt est toujours 0, donc, sur un volume FAT32, RootDirSectors vaut toujours 0. Le 32 ci dessus est la taille d'une entrée de répertoire FAT en octets. Notez également que ce calcule arrondi par excès.

Le début de la zone de données, le premier secteur du cluster 0 est calculé comme suit :

```
If(BPB_FATSz16 != 0)
FATSz = BPB_FATSz16;
Else
FATSz = BPB_FATSz32;
FirstDataSector = BPB_ResvdSecCnt + (BPB_NumFATs * FATSz) + RootDirSectors;
```

NOTE : Ce numéro de secteur est relatif au premier secteur du volume qui contient le BPB (le secteur qui contient le BPB est numéroté 0). Cela ne cartographie pas forcément directement sur le lecteur car le secteur 0 du volume n'est pas nécessairement le secteur 0 du lecteur en raison du partitionnement.

Soit un quelconque cluster de données valide numéroté N, le numéro du premier secteur de ce cluster (encore une fois, par rapport au secteur 0 du volume) est calculé comme suit :

```
FirstSectorofCluster = ((N – 2) * BPB_SecPerClus) + FirstDataSector;
```

NOTE : Comme BPB_SecPerClus est restreint à des puissances de 2 (1, 2, 4, 8, 16, 32, …), les divisions et les multiplications par BPB_SecPerClus peuvent être réalisées par des opérations de décalage (SHIFT) ; sur les architectures travaillant en complément à 2, ces opérations sont plus rapides que des instructions MULT ou DIV. Sur les processeurs x86 actuels, cela n'est pas vrai car les instructions machine MULT et DIV sont fortement optimisées pour la multiplication et la division par des puissances de 2.

Détermination du type de FAT

Il y a une confusion générale sur le fonctionnement de cette détermination, ce qui conduit à des erreurs plus ou moins fréquentes. Cela fonctionne en fait de manière très simple. Le type de FAT (FAT12, FAT16 ou FAT32) est déterminé par le nombre de clusters sur le volume et *rien* d'autre.

Veuillez lire toute cette section attentivement, tous les mots sont importants. Par exemple, notez que l'affirmation était 'nombre de clusters'. Ce n'est pas la même chose que 'nombre maximum valide de clusters', parce que le premier cluster de données est 2 et pas 1 ou 0.

Pour commencer, regardons comment le 'nombre de clusters' est déterminé. Pour cela, on utilise les champs BPB du volume. D'abord, nous déterminons le nombre de secteurs occupé par le répertoire racine, comme noté ci dessus.

```
RootDirSectors = ((BPB_RootEntCnt * 32) + (BPB_BytsPerSec – 1)) / BPB_BytsPerSec;
```

Notez que sur un volume FAT32, la valeur BPB_RootEntCnt est toujours 0 ; donc sur un volume FAT33, RootDirSectors est toujours 0.

Puis, déterminons le nombre de secteurs dans la zone de données du volume :

```
If(BPB_FATSz16 != 0)
FATSz = BPB_FATSz16;
Else
FATSz = BPB_FATSz32;
If(BPB_TotSec16 != 0)
TotSec = BPB_TotSec16;
Else
TotSec = BPB_TotSec32;
DataSec = TotSec – (BPB_ResvdSecCnt + (BPB_NumFATs * FATSz) + RootDirSectors);
```

Déterminons maintenant le nombre de clusters :

```
CountofClusters = DataSec / BPB_SecPerClus;
```

Veuillez noter que ce calcul arrondi par *défaut*.

Nous pouvons maintenant déterminer le type de FAT. *Lisez attentivement ou vous commettrez une erreur dans un cas !*

Dans l'exemple suivant, quand nous disons '<' cela ne veut pas dire '<='. Notez également que les nombres sont corrects, le premier nombre pour la FAT12 est 4085 ; le second pour la FAT16 est 65525. Ces nombres et le signe '<' ne sont pas faux.

```
If(CountofClusters < 4085) {
/* Volume is FAT12 */
} else if(CountofClusters < 65525) {
/* Volume is FAT16 */
} else {
/* Volume is FAT32 */
}
```

C'est la seule et unique façon de déterminer le type de FAT. Il n'y a aucun volume FAT12 qui peut avoir plus de 4085 clusters. Il n'y a aucun volume FAT16 qui a moins de 4085 clusters ou plus de 65524. Il n'y a aucun volume FAT32 qui a moins de 65525 clusters. Si vous essayez de créer un volume qui violerait cette règle, les systèmes Microsoft ne le gèreraient pas correctement parce qu'ils penseront que le volume a un type de FAT différent de celui avec lequel vous l'avez créé.

NOTE : Comme cela a été signalé plusieurs fois auparavant, il y a beaucoup de code FAT faux qui circule. Il y a beaucoup de code de détermination qui est faux dans 1, 2, 8, 10 ou 16 cas. Pour cette raison, il est conseillé lorsque vous formatez un volume qui a une compatibilité avec le plus grand nombre de code, vous devriez éviter de créer des volumes qui ont près de 4085 ou 65525 clusters. Restez à au moins 16 clusters de chaque coté de cette limite.

Notez également que la valeur CountofClusters est exactement le *nombre* de clusters de données à partir du cluster 2. Le nombre maximum valide de clusters est CountofClusters + 1 et le 'nombre de clusters incluant les deux clusters réservés' est CountofClusters + 2.

Il y a un autre calcul important relatif à la FAT. Soit un numéro de cluster valide quelconque N, ou est l'entrée pour ce cluster dans la FAT pour ce numéro de cluster ? Le seul type de FAT pour lequel ce calcul est compliqué est la FAT12. Pour les FAT16 et FAT32, il est très simple :

```
If(BPB_FATSz16 != 0)
FATSz = BPB_FATSz16;
Else
FATSz = BPB_FATSz32;
If(FATType == FAT16)
FATOffset = N * 2;
Else if (FATType == FAT32)
FATOffset = N * 4;
ThisFATSecNum = BPB_ResvdSecCnt + (FATOffset / BPB_BytsPerSec);
ThisFATEntOffset = REM(FATOffset / BPB_BytsPerSec);
```

ThisFATSecNum est le numéro du secteur de la première FAT qui contient l'entrée pour le cluster N. Si vous voulez son numéro dans la seconde FAT, vous devez ajouter FATSz à ThisFATSecNum ; pour la troisième FAT, 2*FATSz, etc.

Vous pouvez maintenant lire le secteur numéro ThisFATSecNum (souvenez vous que c'est un secteur relatif au secteur 0 du volume FAT). Supposons qu'il est lu dans un tableau d'octets nommé SecBuff. Supposons également que le type WORD est un entier non signé de 16 bits et que le type DWORD est un entier non signé 32 bits.

```
If(FATType == FAT16)
FAT16ClusEntryVal = *((WORD *) &SecBuff[ThisFATEntOffset]);
Else
FAT32ClusEntryVal = (*((DWORD *) &SecBuff[ThisFATEntOffset])) & 0x0FFFFFFF;
```

Récupère le contenu du cluster. Pour définir le contenu de ce cluster, procédez comme suit :

```
If(FATType == FAT16)
*((WORD *) &SecBuff[ThisFATEntOffset]) = FAT16ClusEntryVal;
Else {
FAT32ClusEntryVal = FAT32ClusEntryVal & 0x0FFFFFFF;
*((DWORD *) &SecBuff[ThisFATEntOffset]) =
(*((DWORD *) &SecBuff[ThisFATEntOffset])) & 0xF0000000;
*((DWORD *) &SecBuff[ThisFATEntOffset]) =
(*((DWORD *) &SecBuff[ThisFATEntOffset])) | FAT32ClusEntryVal;
}
```

Observez comment fonctionne le code pour la FAT32. Une entrée FAT32 est en fait une entrée de 28 bits. Les 4 bits de poids fort d'une entrée FAT32 sont réservés. Le seul moment où les 4 bits de poids fort d'une entrée FAT devraient être changés est lors du formatage, quand les 32 bits de l'entrée devraient être mis à 0, y compris les 4 bits de poids fort.

Un peu plus d'explication est de rigueur ici, car cette remarque sur les entrées FAT32 semble être la source de beaucoup de confusions. A la base, les entrées 32 bits de la FAT ne sont pas réellement des valeurs 32 bits mais 28 bits. Par exemple, toutes ces valeurs 32 bits 0x10000000, 0xF0000000 et 0x00000000 indiquent que le cluster est LIBRE, car vous ignorez les 4 bits de poids fort lorsque vous lisez l'entrée du cluster. Si la valeur du cluster libre est 0x30000000 et que vous voulez marquer le cluster comme invalide en stockant la valeur 0x0FFFFFF7 dedans, l'entrée 32 bits contiendra alors 0x3FFFFFF7 car vous devez conserver les 4 bits de poids fort.

Notez que comme BPB_BytsPerSec est toujours divisible par 2 et 4, vous n'aurez jamais à vous inquiéter de l'arrondi d'une entrée de FAT32 ou FAT16 (ce n'est pas le cas en FAT12).

Le code pour la FAT12 est plus compliqué car il y a 1,5 octets (12 bits) par entrée dans la FAT.

```
if (FATType == FAT12)
FATOffset = N + (N / 2);
/* multiplie par 1.5 sans virgule flottante, la division par 2 arrondit par DEFAUT */
ThisFATSecNum = BPB_ResvdSecCnt + (FATOffset / BPB_BytsPerSec);
ThisFATEntOffset = REM(FATOffset / BPB_BytsPerSec);
```

Nous devons maintenant vérifier le cas ou nous devons arrondir :

```
If(ThisFATEntOffset == (BPB_BytsPerSec - 1)) {
/* Ce cluster est au niveau d'une séparation dans la FAT */
/* Il y a plusieurs stratégies pour gérer cela. Le plus */
/* simple est de toujours charger les secteurs FAT en mémoire */
/* par deux si le volume est FAT12 (si vous voulez charger */
/* le secteur FAT N, vous chargerez également le secteur N+1 */
/* à sa suite en mémoire à moins que le secteur N ne soit le dernier */
/* secteur FAT). Nous supposons que c'est la méthode choisie ici */
/* ce qui rend ce test pour savoir si l'on est au niveau d'une séparation */
/* inutile. */
}
```

Nous pouvons maintenant accéder à l'entrée FAT comme si c'était un WORD, comme nous le faisons pour la FAT16, mais si le numéro de cluster est PAIR, nous ne voulons que les 12 bits de poids faible sur les 16 que nous récupérons ; et si le numéro de cluster est IMPAIR, nous ne voulons que les 12 bits de poids fort sur les 16 que nous récupérons.

```
FAT12ClusEntryVal = *((WORD *) &SecBuff[ThisFATEntOffset]);
If(N & 0x0001)
FAT12ClusEntryVal = FAT12ClusEntryVal >> 4; /* Cluster IMPAIR */
Else
FAT12ClusEntryVal = FAT12ClusEntryVal & 0x0FFF; /* Cluster PAIR */
```

Récupère le contenu du cluster. Pour définir le contenu de ce même cluster, procédez comme suit :

```
If(N & 0x0001) {
FAT12ClusEntryVal = FAT12ClusEntryVal << 4; /* Cluster IMPAIR */
*((WORD *) &SecBuff[ThisFATEntOffset]) =
(*((WORD *) &SecBuff[ThisFATEntOffset])) & 0x000F;
} Else {
FAT12ClusEntryVal = FAT12ClusEntryVal & 0x0FFF; /* Cluster PAIR */
*((WORD *) &SecBuff[ThisFATEntOffset]) =
(*((WORD *) &SecBuff[ThisFATEntOffset])) & 0xF000;
}
*((WORD *) &SecBuff[ThisFATEntOffset]) =
(*((WORD *) &SecBuff[ThisFATEntOffset])) | FAT12ClusEntryVal;
```

NOTE : On suppose que l'opérateur >> insère un bit à 0 dans les 4 bits de poids fort et que l'opérateur << insère un bit à 0 dans les 4 bits de poids faible.

La façon dont les données sur un fichier sont associées à un fichier est la suivante. Dans l'entrée de répertoire, le numéro du premier cluster du fichier est enregistré. Le premier cluster du fichier est la donnée associée à ce numéro de cluster et l'emplacement de cette donnée sur le volume est calculé à partir du numéro de cluster comme décrit ci dessus (calcul de FirstSectorofCluster).

Notez qu'un fichier de longueur 0 (sans données associées), a un numéro de cluster à 0 dans l'entrée de répertoire. L'emplacement d'un cluster dans la FAT (voir plus haut pour ThisFATSecNum et ThisFATEntOffset) contient soit une marque EOC (fin de chaîne de

clusters, End Of Clusterchain) soit le numéro du prochain cluster du fichier. La valeur EOC dépend du type de FAT (en supposant que FATContent est le contenue de l'entrée du cluster dans la FAT pour laquelle on doit vérifier si elle contient la marque EOC) :

```
IsEOF = FALSE;
If(FATType == FAT12) {
If(FATContent >= 0x0FF8)
IsEOF = TRUE;
} else if(FATType == FAT16) {
If(FATContent >= 0xFFF8)
IsEOF = TRUE;
} else if (FATType == FAT32) {
If(FATContent >= 0x0FFFFFF8)
IsEOF = TRUE;
}
```

Notez que le cluster dont l'entrée dans la FAT contient la marque EOC est alloué au fichier et est également le dernier cluster associé au fichier. Les pilotes FAT Microsoft utilisent les valeurs 0x0FFF pour la FAT12, 0xFFFF pour la FAT16 et 0x0FFFFFFF pour la FAT32 quand ils définissent le dernier cluster d'un fichier. Cependant, il y a divers utilitaires pour les systèmes Microsoft qui utilisent d'autres valeurs.

Il y a également une marque spéciale 'CLUSTER CORROMPU'. Tout cluster qui contient la marque 'CLUSTER CORROMPU' dans son entrée FAT est un cluster qui ne devrait pas être placé dans la liste des clusters libres car il est sujet à des erreurs disque. La valeur 'CLUSTER CORROMPU' est 0x0FF7 pour la FAT12, 0xFFF7 pour la FAT16 et 0xFFFFFFF7 pour la FAT32. L'autre point important ici est que ces clusters corrompu sont également des clusters perdus (clusters qui apparaissent comme étant alloués car ils contiennent une valeur différente de 0 mais qui ne font partie d'aucune chaîne d'allocation de fichier. Les utilitaires de récupération disque doivent reconnaître les clusters perdus qui contiennent cette valeur particulière comme des clusters corrompus et ne pas changer le contenu de l'entrée.

NOTE : Il n'est pas possible pour la marque de cluster corrompu d'être un numéro de cluster allouable en FAT12 ou FAT16, mais c'est possible en FAT32. Pour éviter une confusion possible par les utilitaires disques, aucun volume FAT32 ne devrait être configuré de manière à ce que 0x0FFFFFF7 soit un numéro de cluster allouable.

La liste des clusters libres est juste la liste de tous les clusters qui contiennent la valeur 0 dans leur entrée FAT. Notez que cette valeur doit être récupérée comme nous l'avons décrit plus haut, de la même manière que pour un cluster non libre. La liste des clusters libres n'est stockée nulle part sur le volume ; elle doit être créée lorsque le volume est monté en scannant la FAT pour trouver les entrées à 0. Sur les volumes FAT32, le secteur BPB_FSInfo *peut* contenir un décompte correct des clusters libres du volume. Voyez la documentation du secteur FSInfo.

A quoi servent les deux clusters réservés au début de la FAT ? Le premier cluster réservé, FAT[0], contient l'octet BPB_Media dans ses 8 bits de poids faible, et tous les autres bits sont à 1. Par exemple, si la valeur BPB_Media vaut 0xF8, pour la FAT12, FAT[0] = 0xFF8, pour la FAT16, FAT[0] = 0xFFF8 et pour la FAT32, FAT[0] = 0xFFFFFFF8. Le second cluster réservé, FAT[1], est initialisé par le formatage à la valeur de la marque EOC. Sur les volumes FAT12, il n'est pas utilisé et contient uniquement la valeur de la marque EOC. Pour la FAT16 et la FAT32, le pilote peut utiliser les deux bits de poids fort de l'entrée FAT[1] pour des drapeaux d'information sur le volume (tous les autres bits sont laissés à 1). Notez que

l'emplacement des bits est différent sur la FAT16 et la FAT32 car il s'agit des 2 bits de poids fort de l'entrée.

Pour la FAT16 :
```
ClnShutBitMask = 0x8000;
HrdErrBitMask = 0x4000;
```

Pour la FAT32 :
```
ClnShutBitMask = 0x08000000;
HrdErrBitMask = 0x04000000;
```

Bit ClnShutBitMask : si le bit est à 1, le volume est propre, sinon il est sale. Cela signifie que le pilote de système de fichiers n'a pas démonté le volume correctement la dernière fois qu'il a monté le volume. Ce serait une bonne idée d'exécuter les utilitaires de réparation disque scandisk/chkdsk car il pourrait être endommagé.

Bit HrdErrBitMask : si le bit est à 1, aucune erreur de lecture/écriture n'ont été rencontrées, sinon, le système a rencontré une erreur d'E/S sur le volume la dernière fois qu'il a été monté, ce qui peut indiquer que quelques secteurs sont corrompus sur le volume. Ce serait une bonne idée d'exécuter les utilitaires de réparation disque scandisk/chkdsk avec une analyse de surface afin de détecter ces secteurs.

Voici deux autres notes importantes sur la zone FAT du volume :

1. le dernier secteur de la FAT ne fait pas forcément entièrement partie de la FAT. La FAT se termine au numéro de cluster du dernier secteur de la FAT qui correspond à l'entrée pour le cluster numéro CountofClusters + 1 (voir le calcul de CountofClusters plus haut), et cette entrée n'est pas nécessairement à la fin du dernier secteur de la FAT. Les pilotes FAT ne devraient faire aucune supposition quant au contenu du dernier secteur de la FAT après l'entrée CountofClusters + 1. Le code de formatage devrait initialiser les bits suivant cette entrée à 0.
2. La valeur BPBFATSz_16 (ou BPB_FATSz32 pour les volumes FAT32) *peut* être plus grand que nécessaire. En d'autres termes, il peut y avoir des secteurs totalement inutilisés à la fin de chaque FAT dans la zone de FAT du volume. Pour cette raison, le dernier secteur de la FAT est toujours calculé en utilisant la valeur CountofClusters + 1 jamais avec la valeur BPB_FATSz16/32. Les pilotes FAT ne devraient jamais faire de suppositions quand au contenu de ces secteurs FAT 'supplémentaires'. Le code de formatage devrait initialiser les bits de ces secteurs à 0.

Initialisation d'un volume FAT

A ce stade, le lecteur attentif devrait avoir une question très importante. Etant donné que le type de FAT (FAT12, FAT16 ou FAT32) est dépendant du nombre de clusters – et que le nombre de secteurs disponibles dans la zone de données du volume est dépendant de la taille de la FAT – lorsque vous avez affaire à un volume non formaté qui n'a pas encore de BPB, comment déterminer tout cela et calculer les valeurs correctes à mettre dans BPB_FATSz16 ou BPB_FATSz32 ? Les systèmes Microsoft le font grâce à une valeur prédéfinie, plusieurs tableaux et un peu d'arithmétique astucieuse[1].

[1] NdT : clever arithmetic

Les systèmes Microsoft n'utilisent que la FAT12 sur les disquettes. Comme il y a un nombre de formats de disquettes limité, on opère de la façon suivant :

```
"Si c'est une disquette de ce type, le BPB ressemble à ça."
```

Il n'y a pas de calcul dynamique avec la FAT12. Pour les formats FAT12, tous les calculs de BPB_SecPerClus et BPB_FATSz16 ont été faits à la main sur un morceau de papier et enregistrés dans un tableau (en faisant attention, bien sûr, que le nombre total de clusters soit inférieur à 4085). Si votre média est plus grand que 4Mo, ne vous embêtez pas avec la FAT12. Utilisez une valeur de BPB_SecPerClus plus petite comme ça, le volume sera FAT16.

Le reste de cette section est totalement spécifique aux volumes ayant 512 octets par secteur. Vous ne pouvez pas utiliser ces tables, ou l'arithmétique astucieuse, avec des lecteurs ayant une taille de secteur différente. La valeur 'prédéfinie' est une taille de volume qui détermine la limite entre FAT16 et FAT32. Toute taille de volume qui lui est inférieure est en FAT16 et tout volume de taille supérieure est en FAT32. Pour Windows, cette valeur est 512Mo. Tout volume FAT de taille inférieure à 512Mo est en FAT16, tout volume de taille supérieure en FAT32.

Veuillez ne pas tirer de conclusion hâtive ici.

Il y a beaucoup de volumes FAT16 qui font plus de 512Mo. Il y a plusieurs façons de forcer le format en FAT16 plutôt que de le laisser en FAT32, il y a une grande variété de code qui implémente différentes limites. Tout ce dont nous parlons ici est la limite *par défaut* de MS-DOS et Windows sur les volumes qui n'ont pas encore été formatés. Il y a deux tables, une pour la FAT16 et l'autre pour la FAT32. La sélection d'une entrée dans ces tables est basée sur la taille du volume en secteurs de 512 octets (la valeur qui ira dans BPB_TotSec16 ou BPB_TotSec32) et la valeur que définit cette table est la valeur BPB_SecPerClus.

```
struct DSKSZTOSECPERCLUS {
DWORD DiskSize;
BYTE SecPerClusVal;
};
/*
*Voici le tableau pour les lecteurs FAT16. NOTEZ que cette table inclus
* des entrées pour des disques plus grands que 512Mo même si typiquement
* seules les entrées pour les disques < 512 MB sont utilisées.
* La façon dont on accède à cette  table est de chercher la première entrée
* dans la table pour laquelle la taille du disque est inférieure ou égale au
* champ DiskSize de cette entrée. Pour que cette table fonctionne correctement
* BPB_RsvdSecCnt doi être 1, BPB_NumFATs doit être 2 et BPB_RootEntCnt doit
* être 512. Tout changement d'une de ces valeurs requiert le changement
* de la valeur DiskSize de la première entrée sinon le nombre de clusters
* pourrait être trop faible pour la FAT16.
*/
DSKSZTOSECPERCLUS DskTableFAT16 [] = {
{ 8400, 0}, /* disques jusqu'à 4.1 Mo, la valeur 0 pour SecPerClusVal évite une erreur */
{ 32680, 2}, /* disques jusqu'à to 16 Mo, clusters de 1 Ko */
{ 262144, 4}, /* disques jusqu'à 128 Mo, clusters de 2Ko */
{ 524288, 8}, /* disques jusqu'à 256Mo, clusters de 4Ko */
{ 1048576, 16}, /* disques jusqu'à 512Mo, clusters de 8Ko */
/* A partir d'ici, les entrée ne sont utilisées que si la FAT16 est forcée */
{ 2097152, 32}, /* disques jusqu'à 1Go, clusters de 16Ko */
{ 4194304, 64}, /* disques jusqu'à 2 Go, clusters de 32Ko */
{ 0xFFFFFFFF, 0} /* tout disque supérieur à 2Go,
                            la valeur 0 pour SecPerClusVal évite une erreur */
};
```

```
/*
* Voici la table pour les lecteurs FAT32. NOTEZ que cette table inclus
* des entrées pour les disques de taille inférieure à 512Mo, même si
* typiquement, seules les entrées pour les disques >= 512Mo ne sont utilisées.
* La façon dont on accède à cette  table est de chercher la première entrée
* dans la table pour laquelle la taille du disque est inférieure ou égale au
* champ DiskSize de cette entrée. Pour que cette table fonctionne correctement
* BPB_RsvdSecCnt doi être 32 et BPB_NumFATs doit être 2
* Tout changement d'une de ces valeurs requiert le changement
* de la valeur DiskSize de la première entrée sinon le nombre de clusters
* pourrait être trop faible pour la FAT12.
*/
DSKSZTOSECPERCLUS DskTableFAT32 [] = {
{ 66600, 0}, /* disques jusqu'à 32,5Mo, la valeur 0 pour SecPerClusVal évite une erreur */
{ 532480, 1}, /* disques jusqu'à 260Mo, clusters de 0,5Ko */
{ 16777216, 8}, /* disques jusqu'à 8Go, clusters de 4Ko */
{ 33554432, 16}, /* disques jusqu'à 16Go, clusters de 8Ko */
{ 67108864, 32}, /* disques jusqu'à 32Go, clusters de 16Ko */
{ 0xFFFFFFFF, 64}/* disques plus grands que 32Go, clusters de 32Ko */
};
```

Donc, étant donné une taille de disque et un type de FAT, nous avons maintenant une valeur BPB_SecPerClus. La seule chose qui nous reste à faire est de calculer combien de secteurs occupe la FAT pour pouvoir définir BPB_FATSz16 ou BPB_FATSz32. Notez que nous supposons que BPB_RootEntCnt, BPB_RsvdSecCnt et BPB_NumFATs sont correctement renseignés. Nous supposons également que DskSize est la taille du volume pour lequel nous renseignons BPB_TotSec32 ou BPB_TotSec16.

```
RootDirSectors = ((BPB_RootEntCnt * 32) + (BPB_BytsPerSec – 1)) / BPB_BytsPerSec;
TmpVal1 = DskSize – (BPB_ResvdSecCnt + RootDirSectors);
TmpVal2 = (256 * BPB_SecPerClus) + BPB_NumFATs;
If(FATType == FAT32)
        TmpVal2 = TmpVal2 / 2;
FATSz = (TMPVal1 + (TmpVal2 – 1)) / TmpVal2;
If(FATType == FAT32) {
        BPB_FATSz16 = 0;
        BPB_FATSz32 = FATSz;
} else {
        BPB_FATSz16 = LOWORD(FATSz);
/* il n'y a pas de BPB_FATSz32 dans le BPB FAT16 */
}
```

Ne perdez pas trop de temps à comprendre comment fonctionnent ces calculs. Les bases pour ce calcul ont compliquées ; le point important est que c'est de cette façon que font les systèmes Microsoft et que ça marche. Notez cependant que ce calcul ne fonctionne pas parfaitement. Il définira parfois une FATSz qui est jusqu'à deux secteurs trop grande pour la FAT16 et jusqu'à 8 secteurs trop grande pour la FAT32. Il ne calculera jamais une FATSz trop petite par contre. Comme il est correct d'avoir une FATSz qui est trop grande, au prix de la perte de quelques secteurs, le fait que ce calcul soit vraiment très simple compense le fait qu'il pêche par sécurité dans certains cas.

Structure du secteur FSInfo et secteur de boot de sauvegarde

Sur un volume FAT32, la FAT peut être une grande structure de données, contrairement à la FAT16 où elle est limitée à un maximum de 128000 secteurs et la FAT12 où elle est limitée à 6000 secteurs. Pour cette raison, une disposition a été prise pour stocker le dernier nombre de clusters libres 'connu' sur un volume FAT32 de manière à ce qu'il n'ai pas à être recalculé à chaque fois qu'une API est appelée pour savoir quelle quantité d'espace libre il reste sur le volume (comme à la fin du listing d'un répertoire). Le numéro du secteur FSInfo est la valeur

dans le champ BPB_FSInfo ; pour les systèmes Microsoft, elle est toujours définie à 1. voici la structure du secteur FSInfo :

Structure du secteur FSInfo FAT32 et secteur de boot de sauvegarde

Nom	Déplacement	Taille	Description
FSI_LeadSig	0	4	Valeur 0x41615252. Cette signature d'en tête est utilisée pour valider le fait que c'est un secteur FSInfo.
FSI_Reserved1	4	480	Ce champ est réservé pour des extensions futures. Le code de formatage FAT32 devrait toujours initialiser ce champ à 0. Les bits de ce champ ne doivent actuellement pas être utilisés
FSI_StrucSig	484	4	Valeur 0x61417272. Une autre signature.
FSI_Free_Count	488	4	Contient le dernier décompte des clusters libres connu. Si la valeur est 0xFFFFFFFF, alors ce nombre est inconnu et doit être calculé. Toute autre valeur peut être utilisée, sans être nécessairement correcte. Elle devrait être au moins inférieure ou égale au nombre de clusters total.
FSI_Nxt_Free	492	4	C'est une aide pour le pilote FAT. Il indique le numéro de cluster à partir duquel le pilote devrait commencer pour rechercher des clusters libres. Comme une FAT FAT32 est grande, il peut être coûteux en terme de temps, si il y a beaucoup de clusters alloués au début de la FAT, que le pilote commence à rechercher un cluster libre à partir du cluster 2. Typiquement, cette valeur est initialisée au numéro du dernier cluster alloué par le driver. Si la valeur est 0xFFFFFFFF, il n'y a aucune aide et le pilote devra commencer à partir du cluster 2. Toute autre valeur peut être utilisée, mais devrait être vérifiée pour s'assurer que c'est un numéro de cluster valide pour le volume.
FSI_Reserved2	496	12	Ce champ est actuellement réservé pour de futures extensions. Le code de formatage FAT32 devrait toujours initialiser tous les octets de ce champ à 0. Les octets de ce champ ne doivent actuellement pas être utilisés.
FSI_TrailSig	508	4	Valeur 0xAA550000. Cette signature de fin est utilisée pour valider le fait qu'il s'agit d'un secteur FSInfo. Notez que les 2 bits de poids fort de cette valeur – qui se trouvent aux déplacements 511 et 512 en octets – correspondent aux octets de signature utilisés aux mêmes déplacement du secteur 0.

Une autre fonctionnalité des volumes FAT32 qui n'est pas présente sur les FAT12/16 est le champ BPB_BkBootSec. Les volumes FAT12/16 peuvent être totalement perdus si le contenu du secteur 0 est écrasé ou si il est corrompu et ne peut plus être lu. C'est un point de défaillance critique pour les volumes FAT12/16. Le champ BPB_BkBootSec réduit la gravité de ce problème sur les volumes FAT32, il y a une copie de sauvegarde des informations du secteur de boot incluant le BPB du volume commençant au secteur indiqué par ce champ.

Dans le cas où les informations du secteur 0 ont été écrasées, tout ce qu'un utilitaire de réparation disque a à faire est de restaurer le(s) secteur(s) de boot à partir de la sauvegarde. Dans le cas où le secteur 0 est corrompu, cela permet de monter le volume et d'offrir à l'utilisateur la possibilité d'accéder à ses données avant de remplacer le disque.

Ce second cas, quand le secteur 0 est corrompu, est la raison pour laquelle aucune valeur autre que 6 ne devrait être placée dans le champ BPB_BkBootSec. Si le secteur 0 est illisible, plusieurs systèmes d'exploitation sont paramétrés 'en dur' pour vérifier s'il existe un secteur de boot de secours commençant au secteur 6 du volume FAT32. Notez qu'il y a un secteur de boot *complet* commençant au secteur indiqué par BPB_BkBootSec. Le 'secteur de boot' Microsoft FAT32 en fait en fait trois. Il y a une copie de ces trois secteurs à partir du secteur BPB_BkBootSec. Une copie du secteur FSInfo est également présente, même si la valeur du

champ BPB_FSInfo du secteur de boot de sauvegarde est la même que celle du BPB du secteur 0.

NOTE : Ces trois secteurs ont la signature 0xAA55 aux déplacements 510 et 511, comme le premier secteur de boot (voir les explications ci dessus à la fin de la description de la structure BPB).

Structure d'un répertoire FAT

Voici une explication réduite des entrées de répertoires FAT. Ce document ignore totalement la question des noms de fichiers longs et ne parle que des entrées de répertoires courtes. Pour une description plus complète de la structure des répertoires FAT, voyez le document 'FAT : Long Name On-Media Format Specification'.

Un répertore FAT n'est rien d'autre qu'un 'fichier' composé d'une liste linéaire de structures de 32 octets. Le seul répertoire spécial, qui doit toujours être présent, est le répertoire racine. Pour les médias FAT12 et FAT16, le répertoire racine est situé à un emplacement fixe sur le disque qui suit immédiatement la dernière FAT et est de taille fixe en secteurs calculée à partir de la valeur BPB_RootEntCnt (voir les calculs pour RootDirSectors plus haut). Pour les médias FAT12 et FAT16, le premier secteur du répertoire racine est un numéro de secteur relatif au premier secteur du volume FAT :

```
FirstRootDirSecNum = BPB_ResvdSecCnt + (BPB_NumFATs * BPB_FATSz16);
```

Pour la FAT32, le répertoire racine peut être de taille variable et est une chaîne de clusters, comme tout autre répertoire. Le numéro du premier cluster du répertoire racine sur un volume FAT32 est enregistré dans BPB_RootClus. Contrairement aux autres répertoires, le répertoire racine, sur n'importe quel type de FAT, n'a pas d'horodatage, pas de nom de fichier (autre que le nom de fichier implicite '\') et ne contient pas les fichiers '.' et '..' comme deux premières entrées de répertoires. Le seul autre aspect spécial du répertoire racine est que c'est le seul répertoire sur le volume FAT pour lequel il est valide d'avoir un fichier qui n'a que le bit ATTR_VOLUME_ID d'engagé (voir ci dessous).

Structure d'entrée de répertoire FAT 32 octets

Nom	Déplacement	Taille	Description
DIR_Name	0	11	Nom court du fichier.
DIR_Attr	11	1	Attributs de fichiers : ATTR_READ_ONLY 0x01 ATTR_HIDDEN 0x02 ATTR_SYSTEM 0x04 ATTR_VOLUME_ID 0x08 ATTR_DIRECTORY 0x10 ATTR_ARCHIVE 0x20 ATTR_LONG_NAME ATTR_READ_ONLY | ATTR_HIDDEN | ATTR_SYSTEM | ATTR_VOLUME_ID Les deux bits de poids fort de l'octet d'attribut sont réservés et devraient toujours être positionnés à 0 quand un fichier est créé et jamais modifiés ou regardés après ça.
DIR_NTRes	12	1	Réservé pour Windows NT. Positionné à 0 quand un fichier est créé et jamais modifié après ça.
DIR_CrtTimeTenth	13	1	Horodatage en millisecondes lors de la création du fichier. Ce champ contient en fait un compte de dixièmes de secondes. La précision de la partie des secondes de DIR_CrtTime est de 2, donc ce champ est un compteur de dixièmes de secondes et son intervalle de validité est de 0 à 199 inclus.
DIR_CrtTime	14	2	Heure à laquelle le fichier a été créé.
DIR_CrtDate	16	2	Date à laquelle le fichier a été créé.
DIR_LstAccDate	18	2	Date de dernier accès. Notez qu'il n'y a pas d'heure de dernier accès, juste une date. C'est la date de la dernière lecture ou écriture. Dans le cas d'une écriture, elle doit être définie à la même date que DIR_WrtDate.
DIR_FstClusHI	20	2	Mot de poids fort du numéro du premier cluster du fichier (toujours à 0 en FAT16/FAT32).
DIR_WrtTime	22	2	Heure de la dernière écriture. Notez que la création du fichier est considérée comme une écriture
DIR_WrtDate	24	2	Date de la dernière écriture. Même remarque que ci dessus.
DIR_FstClusLO	26	2	Mot de poids faible du numéro du premier cluster du fichier.
DIR_FileSize	28	4	DWORD de 32 bits conservant la taille du fichier en octets

DIR_Name[0]

Quelques notes à propos du premier octet (DIR_Name[0]) d'une entrée de répertoire FAT :

- Si DIR_Name[0] = 0xE5, l'entrée de répertoire est libre (il n'y a pas de nom de fichier ou de répertoire dans cette entrée).
- Si DIR_Name[0] = 0x00, l'entrée de répertoire est libre (comme pour 0xE5) et il n'y a plus d'entrée de répertoire libre après celle ci (tous les octets DIR_Name[0] des entrées suivantes sont également à 0).
  La valeur 0, plutôt que la valeur 0xE5, indique au pilote de système de fichiers FAT que le reste des entrées de ce répertoire n'ont pas besoin d'être examinées puisqu'elles sont toutes libres.
- Si DIR_Name[0] = 0x05, alors la valeur du caractère pour cet octet est 0xE5. 0xE5 est un code de caractre KANJI valide pour le jeu de caractère utilisé au Japon. La valeur 0x05 est utilisée pour que ce nom de fichier spécial pour le Japon puisse être géré correctement et ne fasse pas penser au pilote que l'entrée est libre.

Le champ DIR_Name est en fait dissocié en deux parties : la partie principale du nom sur 8 caractères et l'extension de 3 caractères. Ces deux parties sont 'alignées par des espaces' avec des octets à 0x20.

DIR_Name[0] ne devrait pas valoir 0x20. Il y a un '.' implicite entre la partie principale et l'extension du nom qui n'est pas présent dans DIR_Name. Les caractères en minuscules ne sont pas admis, en effet, ces caractères sont spécifiques au pays.

Les caractères suivants ne sont valides dans aucun octet de DIR_Name :

- Les valeurs inférieures à 0x20 sauf pour le cas spécial de 0x05 dans DIR_Name[0] décrit plus haut.
- 0x22, 0x2A, 0x2B, 0x2C, 0x2E, 0x2F, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x5B, 0x5C, 0x5D et 0x7C.

Voici quelques exemples sur la façon dont les noms entrés par l'utilisateur sont stockés dans DIR_Name :

```
"foo.bar" -> "FOO BAR"
"FOO.BAR" -> "FOO BAR"
"Foo.Bar" -> "FOO BAR"
"foo" -> "FOO "
"foo." -> "FOO "
"PICKLE.A" -> "PICKLE A "
"prettybg.big" -> "PRETTYBGBIG"
".big" -> illégal, DIR_Name[0] ne peut pas être 0x20
```

Dans les répertoires FAT, tous les noms sont uniques. Regardez les trois premiers exemples précédents. Ces différents noms se réfèrent tous au même fichier et il ne peut y avoir qu'un seul fichier avec le DIR_Name défini à 'FOO BAR' dans un même répertoire.

DIR_Attr spécifie les attributs du fichier :

ATTR_READ_ONLY	indique que l'écriture sur le fichier doit échouer
ATTR_HIDDEN	indique qu'un listing de répertoire normal ne doit pas afficher ce fichier
ATTR_SYSTEM	indique que ce fichier est un fichier du système d'exploitation
ATTR_VOLUME_ID	il ne doit y avoir qu'un seul 'fichier' sur le volume qui a cet attribut de définit, et ce fichier doit être le répertoire racine. Le nom de ce fichier est en fait l'étiquette du volume. DIR_FstClusHI et DIR_FstClusLO doivent toujours valoir 0 pour l'étiquette du volume (aucun cluster de données n'est alloué au fichier d'étiquette de volume).
ATTR_DIRECTORY	indique que le fichier contient en fait d'autres fichiers
ATTR_ARCHIVE	Cet attribut aide les utilitaires de sauvegarde. Ce bit est engagé par le pilote FAT lorsqu'un fichier est créé, renommé ou écrit. Les utilitaires de sauvegarde devraient utiliser cet attribut pour indiquer quels fichiers sur le volume ont été modifiés depuis la dernière fois où l'utilitaire a été lancé.

Notez que la combinaison de bits d'attributs ATTR_LONG_NAME indique que le 'fichier' fait en fait partie de l'entrée de nom long pour un autre fichier. Voyez la spécification des noms de fichiers longs FAT pour plus d'information sur cette combinaison d'attributs.

Quand un répertoire est créé, un fichier avec le bit ATTR_DIRECTORY engagé dans le champ DIR_Attr, vous devez positionner sa valeur DIR_FileSize à 0. DIR_FileSize n'est pas utilisé et vaut toujours 0 sur un fichier avec l'attribut ATTR_DIRECTORY (la taille d'un répertoire est déterminée simplement en suivant leur chaîne de clusters jusqu'à la marque EOC). Un cluster est alloué au répertoire (à moins que ce ne soit le répertoire racine sur les FAT12/16) et vous renseignez DIR_FstClusLO et DIR_FstClusHI à ce numéro de cluster et placez une marque EOC dans cette entrée de cluster dans la FAT. Ensuite, vous initialisez tous les bits de ce secteur à 0. Si le répertoire est le répertoire racine, vous avez fini (il n'y a pas d'entrées point ou *pointpoint* dans le répertoire racine). Si le répertoire n'est pas le répertoire racine, vous devez créer deux entrées spéciales dans les deux premières entrées de 32 octets du répertoire (les deux premières entrées de 32 octets de la zone de données du cluster que vous venez d'allouer).

La première entrée de répertoire a DIR_Name positionné à :
"."

La seconde a DIR_Name positionné à :
".."

Ces entrées sont appelées les entrées *point* et *pointpoint*. Le champ DIR_FileSize de ces deux entrées est positionné à 0 et tous les champs de date et d'heure de ces entrées sont initialisés aux mêmes valeurs que celles que vous avez définies pour le répertoire que vous venez de créer. Vous devez maintenant initialiser DIR_FstClusLO et DIR_FstClusHI pour l'entrée *point* (la première entrée) aux mêmes valeurs que celles que vous avez placées dans ces champs pour le répertoire que vous venez de créer (numéro de cluster du cluster qui contient les entrées *point* et *pointpoint*).
Enfin, vous initialisez DIR_FstClusLO et DIR_FstClusHI pour l'entrée *pointpoint* (la seconde entrée) au numéro du premier cluster du répertoire dans lequel vous venez de créer le répertoire (0 si ce répertoire est le répertoire racine, même sur les volumes FAT32).

Voici un résumé sur les entrées *point* et *pointpoint* :

- L'entrée point est une entrée de répertoire qui pointe sur celui qui la contient.
- L'entrée pointpoint pointe sur le premier cluster du répertoire parent de celui qui la contient (qui est 0 si ce répertoire parent est le répertoire racine).

Format de date et d'heure

Beaucoup de systèmes de fichiers FAT ne supportent pas de date et d'heure autres que DIR_WrtTime et DIR_WrtDate. Pour cette raison, DirCrtTimeMil, DIR_CrtTime, DIR_CrtDate et DIR_LstAccDate sont des champs optionnels. Cependant, DIR_WrtTime et DIR_WrtDate *doivent* être supportés. Si les autres champs de date et d'heure ne sont pas gérés, ils doivent être initialisés à 0 lors de la création et ignorés ensuite.

Format de date. Une date d'entrée de répertoire FAT est un champ 16 bits qui est relatif au 1/1/1980. En voici le format (le bit 0 est le bit de poids faible du mot de 16 bits, le bit 15 est le bit de poids fort du mot de 16 bits) :

Bits 0-4 : compteur à 2 secondes, intervalle de 0 à 19 inclus (0 à 58 sec).
Bits 5-10 : minutes, intervalle de 0 à 59 inclus.
Bits 11-15 : heures, intervalle de 0 à 24.

L'intervalle de temps valide va de minuit 00:00:00 à 23:59:58.

Autres notes à propos des répertoires FAT

- Les entrées de répertoire à nom long sont identiques sur tous les types de FAT. Voyez la spécification des noms de fichiers long pour plus de détails.
- DIR_FileSize est un champ 32 bits. Pour les volumes FAT32, le pilote FAT ne doit pas permettre la création d'une chaîne de clusters de plus de 0x100000000 octets et le dernier octet du dernier cluster d'une chaîne de cette longueur ne peut pas être alloué au fichier. Ceci afin qu'aucun fichier ne fasse plus de 0xFFFFFFFF octets. C'est une limite fondamentale sur tous les systèmes de fichiers FAT. La taille maximum autorisée sur un volume FAT est 0xFFFFFFFF (4 294 967 295) octets.
- De même, un pilote FAT ne doit pas permettre à un répertoire (un fichier qui contient en fait d'autres fichiers) d'être plus grand que 65 536 * 32 (2 097 152) octets.

NOTE : cette limite ne s'applique *pas* au nombre de fichiers dans la répertoire. Cette limite est posée sur le répertoire lui même et n'a rien à voir avec le contenu du répertoire. Il y a deux raisons à cette limite :

1. Comme les répertoires FAT ne sont pas triés ou indexés, c'est une mauvaise idée de créer de très grands répertoires ; autrement, les opérations comme la création d'une nouvelle entrée (qui nécessite la vérification de chaque entrée de répertoire allouée pour vérifier que le nom n'existe pas déjà) deviennent très lentes.
2. Il y a beaucoup de pilotes FAT et d'utilitaires disque, y compris ceux de Microsoft, qui s'attendent à pouvoir compter les entrées d'un répertoire en utilisant une variable WORD de 16 bits. Pour cette raison, les répertoires ne peuvent pas avoir pour plus de 16 bits d'entrées.

Conformité aux spécifications

La conformité à ces spécifications est définie en testant le volume sur des systèmes d'exploitation FAT de référence. Les systèmes d'exploitation de référence pour la FAT sont Microsoft Windows 98 et Microsoft Windows 2000 (basé sur la technologie NT).

Votre volume FAT est conforme à ces spécifications si et seulement si les deux systèmes montent le volume, le vérifient avec un outil de vérification disque fournit (chkdsk.exe pour Windows 2000, scandisk.exe pour Windows 98) et ne trouvent aucune erreur. La procédure de base est de créer un volume FAT avec votre système et vos outils et de placer le disque, ou le média dans le cas d'un lecteur amovible, sur un ordinateur exécutant un des systèmes de référence et de le tester.

