SYS2

Système d'exploitation

M.Bastreghi (mba)

Haute École Bruxelles Brabant — École Supérieure d'Informatique

Année académique 2020 / 2021

Avancement

- FAT
- Structure
- Table d'index
- Répertoires
- Intégrité
- Table d'Index - Tailles
- Conclusions

FAT - File Allocation Table

Système de fichiers de DOS (dérivé de CP/M unix), reconnu par d'autres OS :

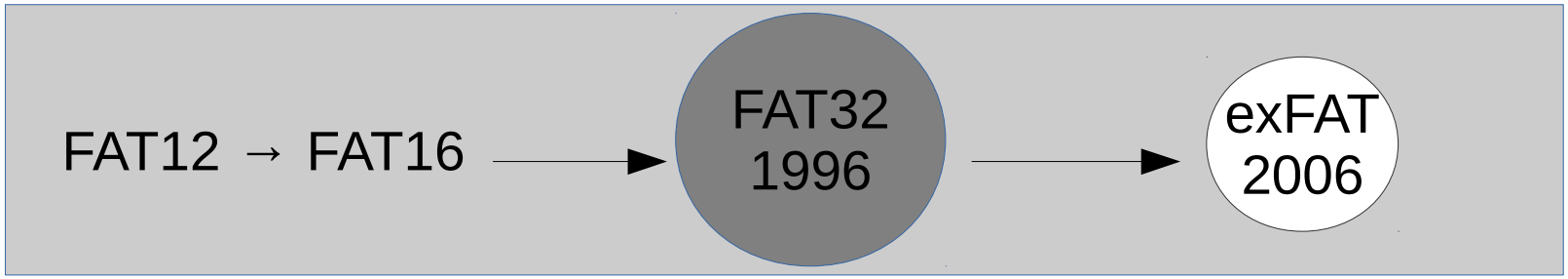
Windows et **linux** ...

documenté :

1. Hardware White Paper : "FAT : General Overview of On-Disk Format"
2. ISO/IEC 9293 :1994

exFAT (extensible FAT) : format propriétaire de Microsoft récent

FAT - 32



FAT - allocation

allocation de l'espace par blocs (clusters)

taille d'un cluster : définie au formatage

- taille des clusters : 2^n secteurs (n valant de 0 à 7)
  - 512B pour n = 0
  - 1024B
  - ...
  - 64KiB pour n = 7

FAT - clusters

taille des clusters en FAT32

Table – Tailles des clusters

n	0	1	2	...	6	7
2^n = nb secteurs/cluster	1	2	4	...	64	128
taille du cluster	512B	1KiB	2KiB	...	32KiB	64KiB

Taille du secteur = 512 Bytes

FAT - clusters

taille des clusters en FAT32

Table – Tailles des clusters

n	0	1	2	...	6	7
2^n = nb secteurs/cluster	1	2	4	...	64	128
taille du cluster	512B	1KiB	2KiB	...	32KiB	64KiB

Taille du secteur = 512 Bytes

- exFAT : n=25 avec une taille maximum de 32MiB pour les clusters(16 -> 32MiB ou ? 25 -> 16GiB ?)
- exFAT permettra des tailles de secteurs de 512 Bytes et 4096 Bytes et des tailles de Cluster allant jusque 32MiB

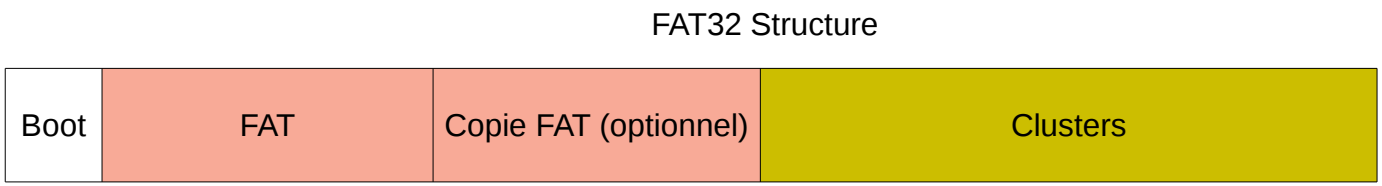
Avancement

- FAT
- Structure
- Table d'index
- Répertoires
- Intégrité
- Table d'Index - Tailles
- Conclusions

Structure - FAT32

structure d'une partition FAT32

métadonnées et données du système de fichiers



FAT16

Une zone supplémentaire de taille fixe entre les tables FAT et la zone des clusters est réservée au répertoire racine.

D'emplacement fixe et calculable le répertoire racine en FAT16 est limité à 256 entrées et son allocation contiguë.

Structure - Secteur de Boot

le **Boot sector** et Bios Parameter Block
métadonnées du système de fichiers :

(32 secteurs pour FAT32)

- (3B) instruction de saut au chargeur du système (plus loin)
- ...
- (2B) taille d'un secteur (512B)
- (1B) nombre de secteurs par cluster
- (2B) taille zone réservée (Boot) (1-32)
- (1B) nombre de FAT (1-2)
- (2B) nombre d'entrées de 32 Bytes dans le répertoire racine en FAT16 (**512**)
- ...
- code du chargeur du système

Structure - Secteur de Boot

le **Boot sector** et Bios Parameter Block
métadonnées du système de fichiers :
(32 secteurs pour FAT32)

- (3B) instruction de saut au chargeur du système (plus loin)
- ...
- (2B) taille d'un secteur (512B)
- (1B) nombre de secteurs par cluster
- (2B) taille zone réservée (Boot) (1-32)
- (1B) nombre de FAT (1-2)
- (2B) nombre d'entrées de 32 Bytes dans le répertoire racine en FAT16 (512)
- ...
- code du chargeur du système

Le secteur de boot contient des métadonnées du système de fichiers et permet la localisation du répertoire racine dont l'emplacement(cluster 2 pour FAT32) est calculable.

Structure - Table FAT

chaînage des blocs par table d'index

La table FAT est un index contenant des numéros de clusters codés sur 4 Bytes (2 Bytes pour FAT16)

- **FAT** File Allocation Table (table = index)
  - FAT32 numéros de clusters codés sur **28 bits**
  - FAT16 numéros de clusters codés sur **16 bits**
  - exFAT numéros de clusters codés sur **32 bits**

chaînage des blocs par table d'index

La table FAT est un index contenant des numéros de clusters codés sur 4 Bytes (2 Bytes pour FAT16)

- **FAT** File Allocation Table (table = index)
  - FAT32 numéros de clusters codés sur **28 bits**
  - FAT16 numéros de clusters codés sur **16 bits**
  - exFAT numéros de clusters codés sur **32 bits**

- en FAT les numéros de clusters dans la table d'index utilisent 2B ou 4B selon le type de FAT
- FAT 12 utilisait 12 bits donc 1,5 bytes pour les numéros de cluster

Structure - Chaînage des clusters

Table d'index :

- L'index est un tableau ayant autant d'entrées qu'il y a de **clusters** dans la partition
- À chaque numéro de clusters est associé, dans l'index, à l'indice correspondant au numéro de cluster, un état : libre, défectueux ou "chaîné".

Dans le cas d'un cluster chaîné, l'index contient le **numéro du cluster suivant** dans le chaînage ou une marque de fin si le cluster est le dernier du fichier.

Structure - Chaînage des clusters

- Une valeur 9 à l'indice 4 veut dire que le cluster 4 est suivi par le cluster 9 ...,4,9,...
- Chaque cluster peut ainsi être suivi par un autre (au sein du fichier). Un cluster n'appartenant qu'au plus à un fichier.

Structure - Chaînage des clusters

- Si un cluster indique son suivant au moyen de la table, chaque fichier peut être reconstitué entièrement à condition de connaître le numéro P du premier cluster du fichier sur disque.
- La taille du fichier ainsi que le premier cluster du fichier sont renseignés dans le répertoire parent

Structure - Racine

FAT32

- racine = chaîne de clusters, **le premier cluster est renseigné dans la zone boot** (souvent le cluster 2)

Le chaînage permet une longueur variable contrairement au répertoire racine de FAT16

Structure - Clusters libres

- Un cluster libre est marqué comme tel (valeur 0) dans l'index. Attribuer un nouveau cluster à un fichier requiert de parcourir l'index.

- Un cluster libre est marqué comme tel (valeur 0) dans l'index. Attribuer un nouveau cluster à un fichier requiert de parcourir l'index.

- On pourrait imaginer de chaîner tous les clusters libres comme si ils appartenaient à un fichier, pour les trouver plus facilement. C'est ce que fera le système de fichiers **ext**.
- la représentation des clusters libres en exFAT ressemble plus à celle adoptée en **ext2** : par une table de bits (**Allocation Bitmap**). Cette approche, combinée avec une **préallocation** de clusters permet de réduire le problème de fragmentation des fichiers.

Structure - Exemple de chaînage

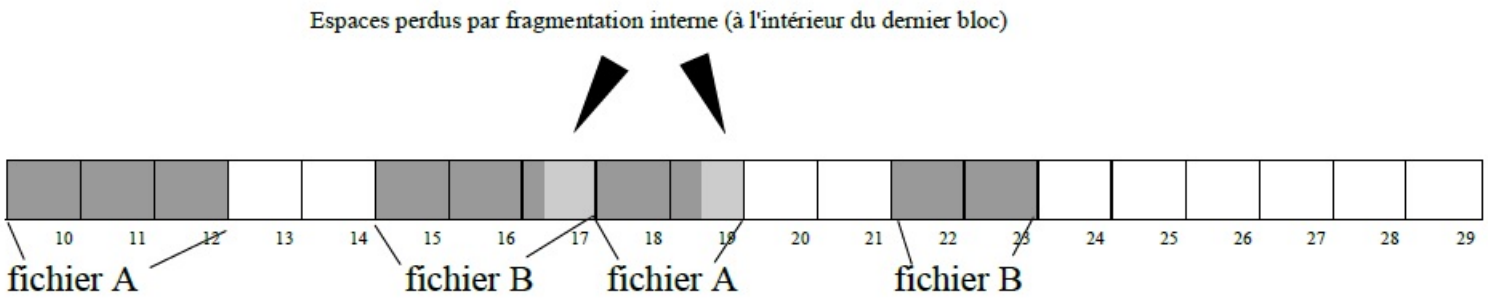


fig 3.2.1.2.1 :blocs sur le disque

Structure - Exemple de chaînage

Reconstituons le fichier A : 10-11-12-18-19

A démarre au cluster 10 (métadonnée du fichier)
à l'indice 10 de l'index on a le cluster suivant de A : 11...
l'indice correspondant à la valeur -1 dans l'index est le dernier cluster du fichier

Avancement

- FAT
- Structure
- Table d'index
- Répertoires
- Intégrité
- Table d'Index - Tailles
- Conclusions

Table d'Index - métadonnées pour le chaînage

Métadonnées nécessaires au chaînage des clusters en FAT :

- La **table d'index** est une métadonnée du système de fichiers
- Le **premier cluster** d'un fichier, renseigné dans le **répertoire parent** est la métadonnée du fichier qui permet de retrouver les clusters suivants du fichier

Table d'index - Valeurs réservées

FAT : certaines valeurs réservées

- 0 - cluster **disponible**
- -1 - **dernier** cluster du fichier
- cluster **défectueux**

Table d'index - Localisation d'un byte

Qu'en est-il de la problématique de la localisation d'un byte du fichier ?
Quelles métadonnées sont nécessaires ? :

métadonnées du système de fichiers :

- taille du secteur
- nombre de secteurs par cluster | taille du cluster
- début du cluster 2
- **table d'index**

métadonnées du fichier :

- cluster associé au **premier bloc** du fichier
- longueur du fichier (et occupation du dernier bloc)

Table d'index - Localiser un byte

localiser le byte N ?

-> traduire "byte N" en "**position** dans le secteur **x**"

1. **F :** F = N DIV TailleBloc
2. P obtenu en suivant le chaînage
3. **adresse** de P = début du cluster 2 + ((P-2) x nbSecteursParCluster)
4. **x** = (adresse de P) + ((N MOD TailleCluster) DIV TailleSecteur)
5. **position du byte N** dans x = N MOD TailleSecteur

Table d'index - Localisation - exemple

Si le cluster 2 est aligné sur le secteur 2048 et un cluster contient deux secteurs (1KiB,1024B)
cherchons le byte N=3600 du **fichier composé des blocs (22-23-15-16-17)**

1. F = 3600B DIV 1KiB = bloc 3 du fichier
2. P(F) = cluster 16 pour (F=3) (22(0)-23(1)-15(2)-16(3)-...)
3. adresse de P = 2048+(16-2) x 2(secteur/bloc) = 2076
4. x = 2076 + (3600B MOD 1024B) DIV 512B = 2077
5. byte N dans le secteur x = 3600B MOD 512B = 16B

le byte 3600 du fichier B est le byte 16 du secteur 2077

Table d'index - Localisation - exemple

Si le cluster 2 est aligné sur le secteur 2048 et un cluster contient deux secteurs (1KiB,1024B)
cherchons le byte N=3600 du **fichier composé des blocs (22-23-15-16-17)**

1. F = 3600B DIV 1KiB = bloc 3 du fichier
2. P(F) = cluster 16 pour (F=3) (22(0)-23(1)-15(2)-16(3)-...)
3. adresse de P = 2048+(16-2) x 2(secteur/bloc) = 2076
4. x = 2076 + (3600B MOD 1024B) DIV 512B = 2077
5. byte N dans le secteur x = 3600B MOD 512B = 16B

le byte 3600 du fichier B est le byte 16 du secteur 2077

les clusters en FAT sont numérotés depuis le n° 2

l'adresse du cluster 16 est donc : P = 2048 + **(16-2)** x 2(secteurs/bloc) = 2076

Questions ?

défi . . .

étant donnée la table d'index suivante (indices dans le bas), des secteurs de 512B et des clusters de 1KiB :

				8		4		10		-1	. . .
		2	3	4	5	6	7	8	9	10	11

localisez le byte 2050 (n°secteur-position) dans le fichier qui démarre au bloc 6 sachant que le cluster 2 est aligné sur le secteur 40000

Avancement

- FAT
- Structure
- Table d'index
- Répertoires
- Intégrité
- Table d'Index - Tailles
- Conclusions

Répertoires

répertoire

Tout fichier/répertoire est décrit dans son répertoire parent sauf la racine

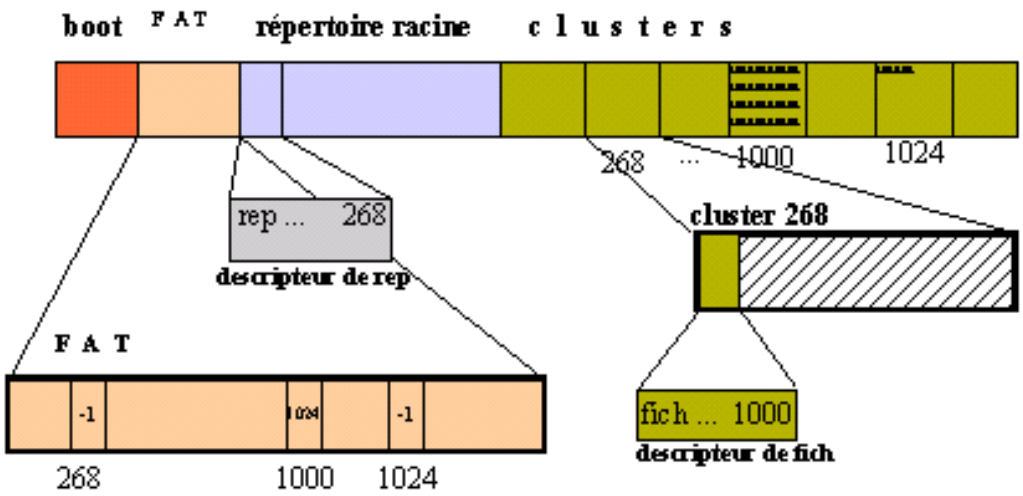
- **fichier d'enregistrements** : descripteurs de fichiers
- **descripteurs** de 32 Bytes (un par fichier/sous-répertoire)
- chaque descripteur contient les **métadonnées d'un fichier sous-répertoire** (notamment **son nom** et son **premier cluster**)
- chaque répertoire sauf la racine contient deux sous-répertoires (. et ..)

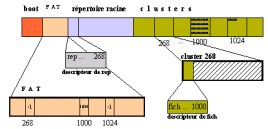
Répertoires - Exemple

Soit une partition FAT16 avec le fichier et le répertoire : /rep/fich
Le fichier contient 1030 caractères 'a'
La taille des clusters est 1KiB (1024 B)
Le répertoire rep est décrit dans le cluster 268
Le fichier fich occupe les clusters 1000 et 1024

Dessinons la structure de ce système de fichiers

Répertoires - Localiser un fichier (FAT16)





- En FAT 32 l'espace réservé à la racine serait remplacé par un chaînage de clusters démarrant au cluster 2
- Les sous-répertoires . et .. du répertoire rep ont été oubliés dans le dessin précédent. Ils devraient précéder l'entrée fich dans le cluster 268 qui décrit ce répertoire.

Répertoires - open

open - comment fait le système pour localiser le fichier /rep/fich ?

- lire les enregistrements successifs du fichier jusque au descripteur de "rep" et obtenir sa position
- lire les enregistrements successifs du fichier rep jusque au descripteur de "fich"
- ajouter la position de fich à l'entrée à la TDFO (Table des Descripteurs de Fichiers Ouverts)

read

- lire les données du fichier fich (clusters chaînés)

Répertoires - open

open - comment fait le système pour localiser le fichier /rep/fich ?

- lire les enregistrements successifs du fichier jusque au descripteur de "rep" et obtenir sa position
- lire les enregistrements successifs du fichier rep jusque au descripteur de "fich"
- ajouter la position de fich à l'entrée à la TDFO (Table des Descripteurs de Fichiers Ouverts)

read

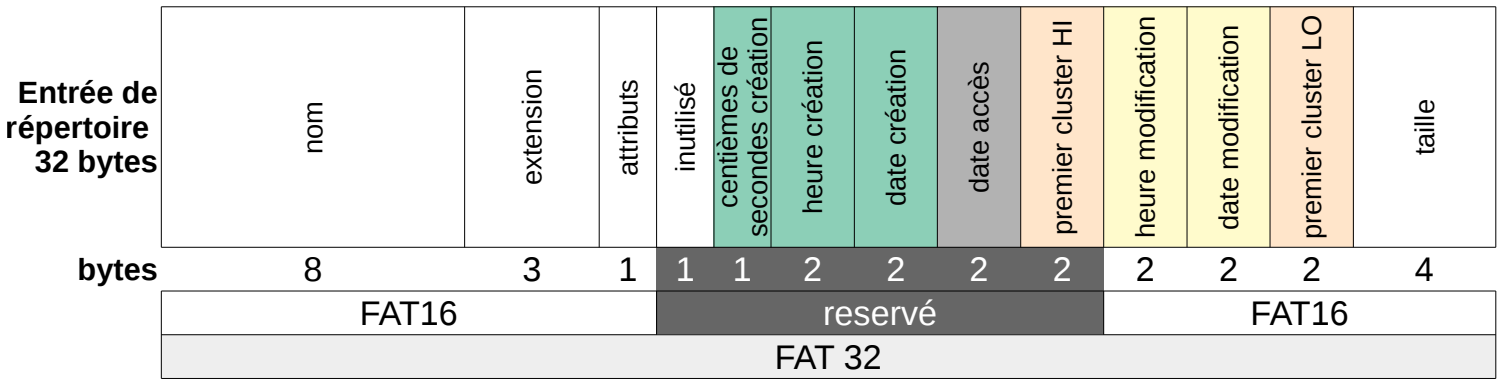
- lire les données du fichier fich (clusters chaînés)

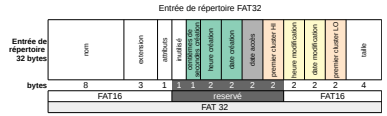
Le Système d'exploitation doit savoir que le cluster 1024 ne contient que 6 'a'

Répertoires - Descripteur

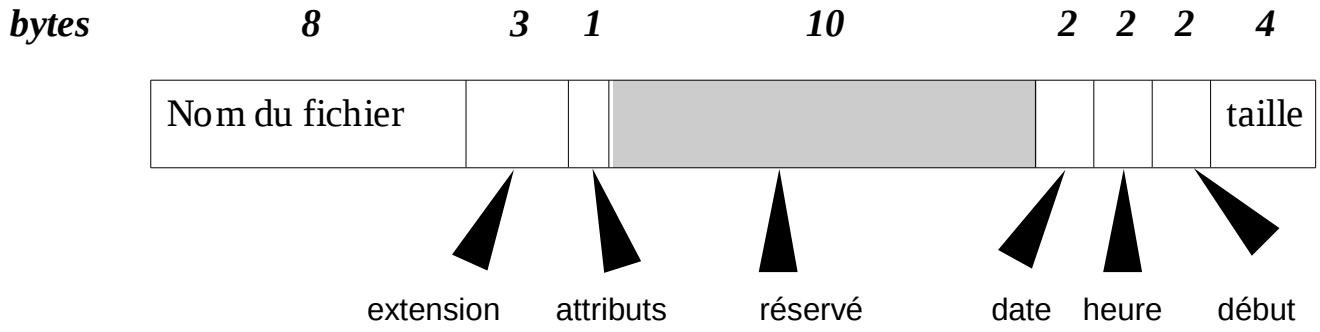
FAT32 détail de l'entrée de répertoire

Entrée de répertoire FAT32

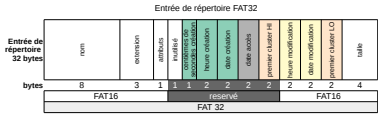




Descripteurs en FAT16 : une partie de 10 bytes (au centre en gris) réservée aux extensions futures et utilisée par FAT32



FAT32 détail de l'entrée de répertoire



- exFAT modifie la structure des entrées de répertoire et, à l'instar de NTFS étend l'utilisation d'entrées de répertoire spécifiques pour les métadonnées du système de fichiers (Allocation Bitmap)

Répertoires - Attributs

attributs :

un par bit

- ARCHIVE
- READ_ONLY
- HIDDEN
- SYSTEM
- DIRECTORY
- VOLUME_ID
- 2 bits à 0 (inutilisés)

Répertoires - Attributs

attributs :

un par bit

- ARCHIVE
- READ_ONLY
- HIDDEN
- SYSTEM
- DIRECTORY
- VOLUME_ID
- 2 bits à 0 (inutilisés)

- FAT est un système conçu pour MS-DOS (système d'exploitation mono utilisateur)
- Les droits sur un système de fichiers FAT se résument à peu de chose.

Répertoires - Noms

nom et extension : 11 Bytes (8+3)

- nom DOS

premier byte du nom : valeurs particulières FAT32

- 0x00 : marque la **fin des entrées**
- 0xF5 : **entrée libre** (fichier supprimé). La valeur 0x05 remplacera la valeur 0xF5 pour représenter ce caractère de l'alphabet Japonais en début de nom :-)

nom et extension : 11 Bytes (8+3)

- nom DOS

premier byte du nom : valeurs particulières FAT32

- 0x00 : marque la **fin des entrées**
- 0xF5 : **entrée libre** (fichier supprimé). La valeur 0x05 remplacera la valeur 0xF5 pour représenter ce caractère de l'alphabet Japonais en début de nom :-)

exFAT
n'utilise plus le premier caractère des noms de fichiers à des fins détournés :

- fin des entrées - Un nouveau champ **type** (un byte) indique, quand il vaut 0, la fin des entrées de répertoire
- entrée libre - Un nouveau bit **inUseField** indique si une entrée est utilisée

Répertoires - Noms longs

- depuis vFAT et FAT32, en cas de nom long on utilise **plusieurs entrées répertoire**
- chaque nouvelle entrée permet d'étendre le nom (+13 caractères)
- l'entrée qui doit être consécutive au nom, est marquée grâce à l'attribut ATTR_LONG_NAME

- depuis vFAT et FAT32, en cas de nom long on utilise **plusieurs entrées répertoire**
- chaque nouvelle entrée permet d'étendre le nom (+13 caractères)
- l'entrée qui doit être consécutive au nom, est marquée grâce à l'attribut ATTR_LONG_NAME

ATTR_LONG_NAME est **ignoré** par DOS
l'attribut ATTR_LONG_NAME correspond à une combinaison inconnue par DOS et donc ignorée

ATTR_LONG_NAME = ATTR_LONG_NAME = READ_ONLY + HIDDEN + SYSTEM + VOLUME_ID

- Les anciens outils, ne verront que la première partie du nom des fichiers
- Surprise ! Pour ceux qui croyaient faire des backups en utilisant de vieux outils

- depuis vFAT et FAT32, en cas de nom long on utilise **plusieurs entrées répertoire**
- chaque nouvelle entrée permet d'étendre le nom (+13 caractères)
- l'entrée qui doit être consécutive au nom, est marquée grâce à l'attribut ATTR_LONG_NAME

exFAT

- Utilise au minimum trois entrées de répertoire par fichier
- Les fichiers dont le nom est > à 15 caractères occupent des entrées supplémentaires, chacune permettant d'ajouter 15 caractères unicode au nom

- depuis vFAT et FAT32, en cas de nom long on utilise **plusieurs entrées répertoire**
- chaque nouvelle entrée permet d'étendre le nom (+13 caractères)
- l'entrée qui doit être consécutive au nom, est marquée grâce à l'attribut ATTR_LONG_NAME

vFAT(16)

La gestion des noms longs combinée avec le nombre réduit d'entrées du répertoire racine permettrait de saturer un système de fichiers vFAT16 avec +/- 30 fichiers de 1 byte !

Répertoires - Heure précision à 2 secondes

heure-date en FAT :

heure et date (2+2) Bytes

- le choix est porté sur une représentation structurée (année, mois, ..)
- si on représentait la date par un nombre de secondes sur 32 bits (4 Bytes), on pourrait représenter sur 4 Bytes 9 années de plus avec une précision supérieure.

heure-date en FAT :

heure et date (2+2) Bytes

- le choix est porté sur une représentation structurée (année, mois, ..)
- si on représentait la date par un nombre de secondes sur 32 bits (4 Bytes), on pourrait représenter sur 4 Bytes 9 années de plus avec une précision supérieure.

- heure : (heure : 5 bits, minutes : 6 bits, secondes : 5 bits) -> max secondes 32 (30)
  précision 2 secondes ! !
  date : (jour : 5 bits, mois : 4 bits, année : 7 bits) -> max année = **127** ans (1980+127 = 2107)
- (2^{32} sec)/(1 jour = 86400 sec) / 365 = **136** ans (1980+136 = 2116)

-> **9 ans de plus avec 1 seconde comme précision au lieu de 2 !**

Répertoires - Précision de l'heure de création FAT32 - exFAT

- à partir de FAT32, un byte supplémentaire permet d'obtenir une précision à la centième de secondes pour l'heure de création, cela grâce à un byte supplémentaire.
- il s'agit d'une valeur comprise entre 0 et 199 avec des **centièmes de secondes** (2 secondes).

Répertoires - Premier cluster FAT32 - exFAT

numéro du premier cluster du fichier (4 Bytes sont nécessaires en FAT32)

C'est donc également l'indice dans la FAT (permettra de localiser tous les clusters suivants)

pour un fichier vide ce premier cluster vaut 0

Répertoires - Premier cluster FAT32 - exFAT

numéro du premier cluster du fichier (4 Bytes sont nécessaires en FAT32)

C'est donc également l'indice dans la FAT (permettra de localiser tous les clusters suivants)

pour un fichier vide ce premier cluster vaut 0

FAT32 utilise 4 Bytes pour les numéros de cluster (deux Bytes supplémentaires par rapport à FAT16. Les bytes de poids fort de ce nombre ont été alloués dans la partie réservée du descripteur (premier cluster HI-LO))

Répertoires - Taille fichier

taille du fichier : 4 Bytes (taille logique)

-> le plus grand fichier sur une partition FAT : **4GiB**

taille du fichier : 4 Bytes (taille logique)

-> le plus grand fichier sur une partition FAT : **4GiB**

exFAT

- Utilise 8 bytes pour une taille théorique de 16PiB
- Cette taille dépassant la taille maximum de la partition exFAT pour des clusters de 32MiB

Répertoires - Taille fichier

taille du fichier : 4 Bytes (taille logique)

-> le plus grand fichier sur une partition FAT : **4GiB**

Les entrées de type répertoire n'ont pas de taille -> suivre la chaîne

Questions ?

questions

- que doit faire le système quand la taille d'un fichier augmente et que le dernier cluster du fichier est plein ?

- que doit faire le système pour créer un nouveau fichier vide ?
- que doit faire le système pour lire un fichier entièrement ?
- pourquoi le premier cluster d'un nouveau répertoire ne vaut pas 0 ?
- que doit faire le système pour créer un nouveau répertoire ?

questions

- les clusters d'une FAT ont tous la même taille [V-F]
- en FAT, il y a perte d'espace disque par fragmentation interne [V-F]
- un répertoire quelconque en FAT tient sur maximum un cluster [V-F]
- la File Allocation Table sert à trouver le premier cluster d'un fichier[V-F]

Répertoires - Grands répertoires

- Localiser un fichier dans un répertoire demande de réaliser des comparaisons
- Pour des répertoires de N entrées, N/2 comparaisons seront faites en moyenne
- Le système de fichiers FAT, ne prévoit pas d'améliorations de performance sur ce point

- Localiser un fichier dans un répertoire demande de réaliser des comparaisons
- Pour des répertoires de N entrées, N/2 comparaisons seront faites en moyenne
- Le système de fichiers FAT, ne prévoit pas d'améliorations de performance sur ce point

Créer un fichier dans un répertoire demande de vérifier que le nom n'existe pas déjà

requiert un parcours complet du répertoire !

- Localiser un fichier dans un répertoire demande de réaliser des comparaisons
- Pour des répertoires de N entrées, N/2 comparaisons seront faites en moyenne
- Le système de fichiers FAT, ne prévoit pas d'améliorations de performance sur ce point

D'autres systèmes de fichiers se penchent sur ce même problème :

- ext2 - répertoires groupés par cylindre
- exFAT - limitation de la taille des répertoires à 512 MiB et utilisation d'un **hash-code** sur deux bytes dans les descripteurs de fichiers (on ne compare que ces deux bytes au lieu du nom en entier)
- NTFS - index sur le noms pour un accès plus direct

Avancement

- FAT
- Structure
- Table d'index
- Répertoires
- Intégrité
- Table d'Index - Tailles
- Conclusions

Intégrité - Incohérences

incohérences

Des états incohérents d'un système de fichier peuvent survenir par exemple suite à une panne de courant
Notamment si celle-ci survient pendant un état instable du système
En cours de modification, par exemple lors d'ajout ou suppression d'un cluster d'un fichier

Intégrité - Ajout

l'ajout d'un cluster demande deux mises à jour :

- le cluster libre pointe vers le cluster suivant du fichier ou la fin du fichier
- le cluster qui précède le nouveau pointe vers le nouveau cluster

Intégrité - Suppression

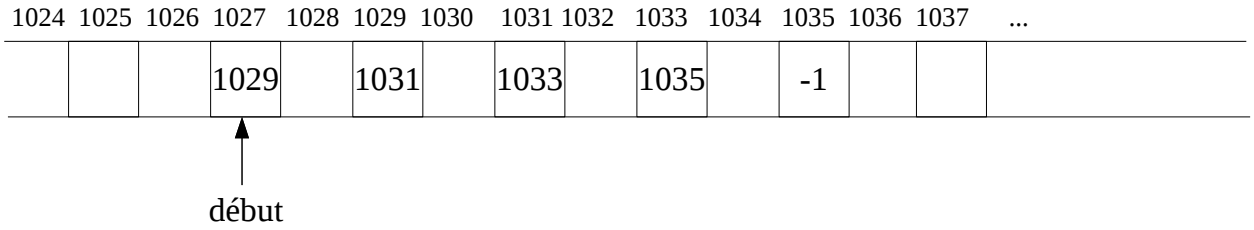
même chose lors de la suppression d'un cluster :

- le cluster qui précède celui à supprimer pointe vers le cluster suivant
- le cluster à supprimer est marqué libre

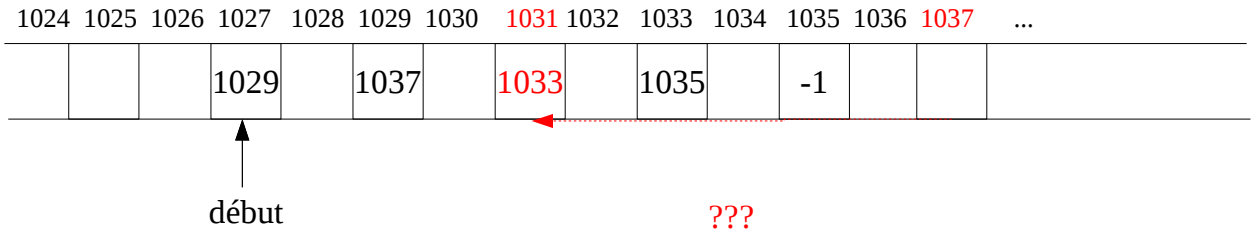
Dans les deux cas, si une panne de courant survient entre les deux mises à jour, la FAT sur disque reste dans un état incohérent

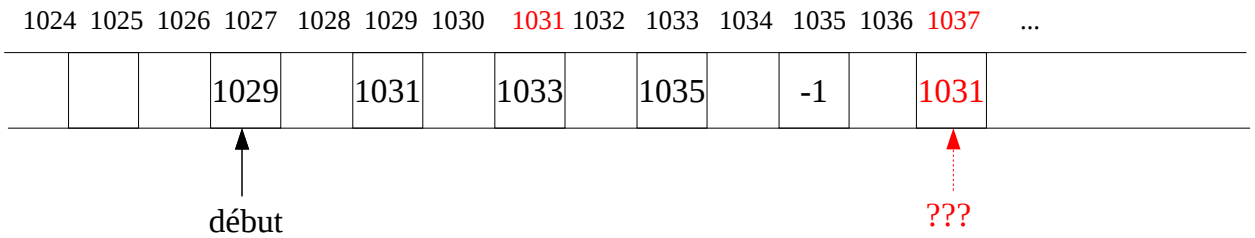
Intégrité - Exemple

cas d'incohérence suite à un ajout :



Une des deux situations suivantes peut survenir après une coupure de courant pendant l'ajout d'un bloc : ici, le bloc 1037 entre les blocs 1029 et 1031





Intégrité - Redondance

Une **incohérence** peut survenir dans une information redondante

- un cluster libre chaîné ?
- un cluster non chaîné, non libre ?

la redondance permet de "réparer" l'incohérence

où est la redondance dans ce cas ?

Intégrité - Réparer ?

une commande comme **chkdsk** utilise cette redondance :

1. parcourt la FAT à la recherche des clusters chaînés : ceux qui appartiennent à un fichier
2. cherche les clusters libres : ceux qui ont un 0 en FAT

en théorie un cluster "normal" doit soit être libre soit être chaîné

La comparaison entre ces deux résultats permet d'isoler des **"chaînes de clusters perdues"**
Ces dernières sont récupérées sous forme de fichiers nommés FILE0000.CHK, FILE0001.CHK dans le répertoire racine

Intégrité - Copie de la FAT

Cluster défectueux ?

cluster défectueux dans un fichier -> perte d'une partie de données

cluster défectueux dans la FAT -> le système de fichiers risque d'être **fortement compromis**

-> la **copie de la FAT** peut être maintenue pour garantir une plus grande fiabilité

si le cluster défectueux est au niveau d'un fichier utilisateur c'est moins grave

Avancement

- FAT
- Structure
- Table d'index
- Répertoires
- Intégrité
- Table d'Index - Tailles
- Conclusions

Table d'Index - - nombre de clusters maximum ?

L'entrée de l'index est un nombre entier (sur 2B ou 4B, ...)

Le plus grand numéro de cluster dépend de la taille de ce nombre entier.

Sur 16 bits on peut représenter les valeurs allant de 0 à 2^{16} -1

Il y a donc au maximum 2^{16} clusters dans une FAT16

À chaque type de FAT correspond un nombre de clusters maximum

Table d'Index - taille partition

En poussant le raisonnement plus loin on peut définir la taille maximum d'un système de fichiers FAT.

Taille Maximum = Nb max clusters x Taille max cluster

Table d'Index - taille partition

Ce qui donne, pour FAT16 et FAT32 :

Table – Tailles maximum théoriques

Nom	bits index	nb clusters	max taille
FAT16	16	2^{16}	4GiB
FAT32	28	2^{28}	16TiB
exFAT	32	2^{32}	128PiB

Table d'Index - taille partition

Ce qui donne, pour FAT16 et FAT32 :

Table – Tailles maximum théoriques

Nom	bits index	nb clusters	max taille
FAT16	16	2^{16}	4GiB
FAT32	28	2^{28}	16TiB
exFAT	32	2^{32}	128PiB

plus grande taille théorique de partition = plus grand nombre de plus grands clusters

en FAT16 = 2^{16} clusters
en FAT32, exFAT = 2^{28} clusters
plus grand cluster = 64 KiB (2^{16}B) et 32MiB (2^{25}B)

pour FAT16 on a : 2^{16} x 2^{16}B = 2^{32}B = 4GiB
pour FAT32 on a : 2^{28} x 2^{16}B = 2^{44}B = 16TiB (conseillé 32GiB)

pour exFAT, avec des clusters de taille 32MiB : 2^{32} x 2^{25}B = 2^{57}B = 128PiB (conseillé 512TiB)

Questions ?

Table d'Index - performance

constat :

- Les clusters d'un même fichier sont éparpillés
- Chaque accès à un nouveau bloc, demande un **double accès disque** (table d'index et bloc de données)

Table d'Index - performance

-> Pour améliorer les performances la table d'index est **chargée en RAM** au démarrage !

(-) Sa **grande taille** cause un ralentissement du système au démarrage (faiblesse de la FAT)

Table d'Index - Taille de l'index

À taille de partition égale, un plus petit cluster donne une plus grande table d'index

quelle taille a l'index d'une FAT ?

- taille index FAT16 = nombre entrées index * 2B
- taille index FAT32 = nombre entrées index * 4B

- nombre entrées index = taille partition / taille clusters

Table d'Index - Taille de l'index

À taille de partition égale, un plus petit cluster donne une plus grande table d'index

quelle taille a l'index d'une FAT ?

- taille index FAT16 = nombre entrées index * 2B
- taille index FAT32 = nombre entrées index * 4B

- nombre entrées index = taille partition / taille clusters

Ce calcul ne tient pas compte de la présence sur la partition de l'index même, il s'agit donc d'une approximation

Table d'Index - Taille de l'index

La taille de la table d'index pour une partition FAT32 de 500GiB avec des clusters de taille 4KiB est **500 MiB** !

avec des clusters de 16KiB on réduit l'index à 125MiB

La taille de la table d'index pour une partition FAT32 de 500GiB avec des clusters de taille 4KiB est **500 MiB** !

avec des clusters de 16KiB on réduit l'index à 125MiB

une partition FAT32 de 500 GiB utilisant des clusters de 4KiB a besoin de 500GiB/4KiB entrées de table pour décrire chacun de ses clusters
s'agissant d'une FAT32, les entrées de table font 4B donc la taille de la table d'index peut être calculée par :

500GiB/4KiB x 4B = 500MiB

Table d'Index - Tailles Maximum

Une table d'index peut devenir très grosse

- la taille maximum d'une table FAT16 est 128KiB
- la taille maximum d'une table FAT32 est **1GiB**

Une table d'index peut devenir très grosse

- la taille maximum d'une table FAT16 est 128KiB
- la taille maximum d'une table FAT32 est **1GiB**

en FAT16 la taille des entrées de l'index est 16 bits - 2B
en FAT32 la taille des entrées de l'index est 28 bits - 4B

la plus grande table d'index est celle qui a le nombre maximum d'entrées
cela se vérifie pour le plus grand nombre de clusters en FAT16 = 2^{16} clusters
cela se vérifie pour le plus grand nombre de clusters en FAT32 = 2^{28} clusters
plus grand index de la FAT16 - 2^{16} x 2B = 2^{17} B = 128 KiB
plus grand index de la FAT32 - 2^{28} x 4B = 2^{30} B = 1 GiB

Table d'Index - Tailles Maximum

Une table d'index peut devenir très grosse

- la taille maximum d'une table FAT16 est 128KiB
- la taille maximum d'une table FAT32 est **1GiB**

exFAT utilise 32 bits pour l'index

Table d'Index - choisir le type de FAT

Quel choix pour une partition de 500 GiB ?
Que dire de la taille des clusters ? Quelle taille aura l'index ?

Quel choix pour une partition de 500 GiB ?
Que dire de la taille des clusters ? Quelle taille aura l'index ?

Avec des très grands clusters on augmente la fragmentation interne

Avec des très petits clusters on augmente la taille de l'index

Table d'Index - critique

La table d'index FAT32 qui est **chargée en RAM** au démarrage du système peut être très grosse ! !

-> Un grand volume FAT32 **ralentit le démarrage du système**

Table d'Index - critique

La table d'index FAT32 qui est **chargée en RAM** au démarrage du système peut être très grosse ! !

-> Un grand volume FAT32 **ralentit le démarrage du système**

Pour cette raison les outils de Microsoft dissuadent de créer des partitions FAT32 de taille > à 32 GiB

exFAT

- exFAT permettra des tailles de cluster allant jusqu'à 32MiB
- conçu pour des fichiers de grande taille permet d'améliorer les performances et réduire la taille de la table
- Quid de la fragmentation interne ?

Questions ?

Table d'Index - Exercice

calculer :

- taille de la **table** FAT avec clusters de 4KiB pour 32 GiB en FAT32(28) ?

défi ...

calculer :

- taille minimum de **clusters** pour 2 TiB en FAT32(28) ?
- taille de la **table** FAT dans ce cas ?

Conclusions - FAT, exFAT

Non multi utilisateur et **Table d'Index** chargée en RAM

- (+)Clés USB, disques externes, cartes pour du multimédia formatés en FAT ou exFAT
- (-)Équilibre entre **taille Index** et **fragmentation interne** à trouver

Conclusions - FAT, exFAT

Non multi utilisateur et **Table d'Index** chargée en RAM

- (+)Clés USB, disques externes, cartes pour du multimédia formatés en FAT ou exFAT
- (-)Équilibre entre **taille Index** et **fragmentation interne** à trouver

le chargement de l'index en RAM **ralentit** le montage du système de fichiers

Conclusions - FAT32

- (++)système de fichiers simple et reconnu par plusieurs systèmes d'exploitation
- (+)utilise l'espace plus efficacement que FAT16
- (-)taille des fichiers **limitée à 4GiB**
- (-)taille théorique des partitions **limitée à 16TiB** 32GiB serait raisonnable ?
- (-)performance pour grands répertoires ?
- (-)fragmentation des fichiers

Conclusions - FAT32

- (++)système de fichiers simple et reconnu par plusieurs systèmes d'exploitation
- (+)utilise l'espace plus efficacement que FAT16
- (-)taille des fichiers **limitée à 4GiB**
- (-)taille théorique des partitions **limitée à 16TiB** 32GiB serait raisonnable ?
- (-)performance pour grands répertoires ?
- (-)fragmentation des fichiers

- La fragmentation de fichiers est due notamment à la manière d'allouer les nouveaux clusters aux fichiers (en début de volume)

Conclusions - exFAT

- (+)exFAT meilleures performances que FAT32 (gros blocs, grands répertoires)
- (+)exFAT meilleure gestion de la fragmentation de fichiers que FAT32 (pluis gros clusters, Allocation Bitmap et pré allocation de clusters aux fichiers)
- (-)exFAT fragmentation interne due aux plus gros clusters et la pré allocation
- (-)exFAT perte de place dans la représentation des répertoires (3 entrées minimum)

Questions ?

question

- Imaginons le travail de l'appel système **open** pour une FAT ?

remerciements

remerciements à P.Bettens et M.Codutti
pour la mise en page :-) Mba

Crédits

Ces slides sont le support pour la présentation orale de l'activité d'apprentissage **SYS2** à la HE2B-ÉSI

Crédits Crédits

La distribution opensuse
du système d'exploitation GNU Linux.
LaTeX/Beamer comme système d'édition.
GNU make, rubber, pdfnup, ... pour les petites tâches.

Images et icônes

deviantart, flickr, The Noun Project