CATALOGUES ET FICHIERS

L'objectif de cet article est de vous permettre de pouvoir bidouiller catalogues et fichiers à votre guise ! Je vois que le petit blond à lunettes a déjà son Disco en main ; il a raison, c'est l'outil idéal pour découvrir les entrailles de vos disquettes. Mais attention, utilisez impérativement une disquette de travail pour vos essais, nous ne saurions être tenus responsable de la perte de vos données ! Bref, commençons.

LE CATALOGUE

Au menu : comment créer des fichiers factices sur une disquette.

« Mais à quoi ça sert ? » s'écrie notre petit blond à lunettes. Eh bien, par exemple, pour marquer des secteurs endommagés (ce que des logiciels comme **MarkError** font automatiquement), pour protéger une zone dans un format personnalisé (pour votre trackloader ou autre), ou encore faire du « catalogue art ».

LES FORMATS

Lorsque l'on veut manipuler le catalogue, la première chose à savoir, c'est où il se trouve sur la disquette. Sur CPC, **l'AMSDOS** nous propose deux formats principaux : le format DATA et le format **SYSTEM** (ou **VENDOR**, c'est pareil).
En format **DATA**, le catalogue est stocké en piste 0 et les secteurs sont numérotés de **&C1** à **&C9** (9 secteurs par piste), alors qu'en format **SYSTEM** ou **VENDOR**, celui-ci est stocké en piste 2 et les secteurs sont numérotés de **&41** à **&49** (toujours 9 secteurs par pistes). Dans ce dernier format, les piste 0 et 1 sont en effet réservées pour mettre des données bootables (qui sont chargées avec la **RSX |CPM**), la seule différence entre les formats **VENDOR** et **SYSTEM** étant que ce dernier contient l'amorce pour le CP/M (le système, CQFD). Ces formats sont tous en 40 pistes, numérotées de 0 à 39, et chaque secteur fait 512 octets ; on comprend ainsi pourquoi une disquette **DATA** offre **178Ko** contre seulement **169Ko** pour une disquette **SYSTEM** ou **VENDOR** qui a deux pistes en moins pour les données utilisateur.

Alors, bien sûr, il existe tout un tas d'autres formats, notamment pour gérer les disquettes 3"1/2, mais le principe reste généralement le même, seuls diffèrent le nombre de pistes, le nom des secteurs, voire la taille des blocs et du catalogue.

ET APRÈS ?

Maintenant, imaginons que vous avez la liste des secteurs que vous voulez allouer. Nous allons donc voir comment se structure le catalogue afin de pouvoir créer un fichier factice (ou non) qui encapsulera ces petits secteurs. En effet, la présence d'un fichier utilisant nos secteurs implique que l'**AMSDOS** n'ira jamais les titiller lorsque vous ferez des manipulations sur votre disquettes (sauvegardes d'autres fichiers) ; c'est notamment très pratique pour verrouiller des secteur endommagés, empêchant ainsi **l'AMSDOS** de les utiliser pour vos précieuses données ! Le catalogue fait **2Ko** et commence sur le premier secteur de la première piste, soit en piste **&00** sur le secteur **&C1** pour se terminer quatre secteurs plus loin, en **&C4**. Pourquoi quatre secteurs plus loin ? Eh bien tout simplement car comme un secteur fait **512** octets, il en faut quatre pour couvrir les **2Ko** de notre catalogue !

Catalogue	
Octet	**Signification**
0-31	1ère entrée de fichier
32-63	2ème entrée de fichier
64-95	3ème entrée de fichier
96-127	4ème entrée de fichier
...	...
2016-2047	64ème entrée de fichier

Or donc notre catalogue se compose d'entrées, chaque entrée référençant un fichier à l'aide d'une séquence de 32 octets dont la signification est la suivante :

Entrée de fichier	
0	**Numéro de "user" du fichier.** Il s'agit tout bonnement du "user" correspondant à la commande **|USER** de **l'AMSDOS** pour lequel votre fichier sera visible, mettez **0** si vous voulez un fichier de base, une valeur entre **1** et **15** pour un autre user, une valeur supérieure à **15** pour un fichier "invisible" vu que les users supérieurs à **15** ne sont pas accessibles via la commande |USER (en bref, mettez ce que vous voulez sauf &E5 qui est réservé pour les fichiers effacés)
1-8	**Nom du fichier.**
9-11	**Extension du fichier.**
12	**Numéro de l'entrée pour ce fichier.** Un fichier de plus de **16**Ko a plusieurs entrées dans le catalogue, elles sont numérotées à l'aide de ce champ.
13-14	**Inutilisé.**

15	**Nombre d'enregistrements de ce fichier.** Un enregistrement fait **128** octets. Cette valeur sert à la fois à savoir combien de blocs existent pour le fichier, mais également quelle taille maximale devra être chargée dans le cas d'un fichier **ASCII** (voir plus bas pour plus de détails sur les types de fichiers).
16-31	**Liste des blocs alloués pour ce fichier.** Un bloc fait **1ko,** soit 8 enregistrements ou encore 2 secteurs. Sachant que seulement **16** blocs peuvent être référencés par entrée et qu'un bloc fait **1Ko,** vous en déduirez logiquement que les fichiers de plus de **16Ko** utilisent plusieurs entrées. Le numéro d'un bloc se calcule simplement à partir du numéro de piste et de secteur : $$bloc=\frac{((piste*9)+secteur)}{2}$$ Attention : -> “secteur” est le numéro de secteur et non son nom, c'est-à-dire que le secteur **&C1** est le numéro 0, de même que le **&41** . -> “piste” est le numéro de piste est relatif à la piste hébergeant le catalogue ; c'est-à-dire que sur un format **DATA** le numéro de piste 0 est effectivement la piste 0, alors que ça sera la piste 2 pour un format **VENDOR** ou **SYSTEM.** À noter : **l'AMSDOS** permet de déplacer un catalogue sur une disquette, on pourrait parfaitement imaginer un catalogue en piste **23.** Ici encore, le numéro de piste utilisé pour le calcul des blocs est relatif à la piste du catalogue, et le numéro de piste 0 correspondra à la piste **23** !

UN EXEMPLE S'IL VOUS PLAÎT !

Du calme, ça arrive. Prenons un cas « complexe ». Imaginons une disquette sur laquelle on veut verrouiller les secteurs suivants :

- Piste 2 : secteurs **&C1** et **&C2**
- Piste 5 : secteur **&C1**
- Piste 9 : secteur **&C6**

Déjà, essayons de voir le nombre de blocs dont nous auront besoin. Comme notre disquette est au format **DATA**, la numérotation des pistes commence à 0.

- Pour le **&C1** secteur piste 2 : bloc = (2*9+'0') / 2 = 9 soit le bloc 9 (**&09**)
- Pour le **&C2** secteur piste 2 : bloc = (2*9+'1') / 2 = 9,5 soit le bloc 9 (**&09**)
- Pour le **&C1** secteur piste 5 : bloc = (5*9+'0') / 2 = 22,5 soit le bloc 22 (**&16**)
- Pour le **&C6** secteur piste 9 : bloc = (9*9+'5') / 2 = 43 soit le bloc 43 (**&2B**)

Argh, mais que sont ces chiffres entre quotes !? Eh bien il s'agit tout simplement du numéro du secteur sur la piste. En format **DATA**, '0' est donc le secteur **&C1** , '1' le secteur **&C2** et '5' le secteur **&C6** .

Que constate-t-on ? Qu'il nous faut allouer non pas quatre mais trois blocs pour verrouiller ces quatre secteurs ! Car, en fait, un bloc ne définit pas un secteur mais deux secteurs consécutifs. Ici, les secteurs **&C1** et **&C2** de la piste 2 pourront donc être déclarés dans le même bloc. Par ailleurs, pour la même raison, le secteur précédant le secteur **&C1** de la piste 5 se retrouvera également verrouillé (soit le secteur **&C9** de la piste 4) ; il est perdu.

Pour mieux comprendre la chose, voici comment **l'AMSDOS** va retrouver les secteurs alloués à partir des numéros de blocs :

- Piste = (2*bloc) / 9 = où “piste” sera le nombre de piste à compter à partir de celle hébergeant le catalogue, soit 0 pour le format **DATA** et 2 pour le format **SYSTEM** ou **VENDOR.**
- Secteur1 = ((2*bloc mod9)+1) or *masque* où “masque” sera **&C0** pour le format **DATA** et **&41** pour le format **VENDOR** ou **SYSTEM.**
- Secteur2 = secteur1+1 et si “secteur2” dépasse la fin de piste, on prendra le premier secteur de la piste suivante.

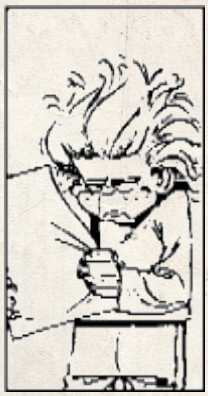
Ceci étant fait, pour remplir notre entrée de fichier, nous avons également besoin de calculer le nombre d'enregistrements :

Enreg = nb blocs*8 = 3*8 = 24 (**&18**)

Voilà, on choisit à présent le nom que l'on va donner à son fichier : “Reserved” pour le nom et “O_o” pour l'extension par exemple. Et puis, pour le “user”, on va mettre un peu ce qu'on veut, mais vous éviterez tout de même la valeur **229** (**&E5**) qui est réservée aux fichiers effacés.

L'avantage d'utiliser un nom de fichier avec des minuscules (ou des caractères spéciaux) est que le **BASIC** et la plupart des outils standard ne seront pas capable de le manipuler ; impossible donc d'effacer notre fichier maison par erreur !

Voici ce que vous devriez finalement obtenir après introduction des données sous l'éditeur de Disco :



&0000	00 52 65 73 65 72 76 65 64 4F 5F 6F 00 00 00 18	. R e s e r v e d O _ o . . . .
&0010	09 16 2B 00 00 00 00 00 00 00 00 00 00 00 00 00	. . + ...
&0020	E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5	. . . .
&0030	Etc...	

Ensuite, par mesure de sécurité, je vous conseille de charger l'explorateur de Disco, et de prendre “Exploration Fichier” afin de vérifier si votre fichier factice est bien là où vous le désiriez.

CAS PARTICULIERS

Mais il y a encore quelques petites précisions à donner sur le catalogue ; pour cela un petit écran de Disco commenté (c'est'y pas beau ça ?).

En effet, je ne vous ai pas parlé des fichiers cachés (dits “système”, ou “SYS”) qui sont invisibles dans le **CAT** du **BASIC** et du **CP/M** ainsi que des fichiers en lecture seule (dits “R/O”) qui apparaissent avec une étoile au **CAT** du **BASIC.** Le codage se fait très simplement au niveau du catalogue et plus précisément au niveau de l'extension du fichier.

Vous aurez peut-être constaté lors de vos manipulations que certains noms de fichiers semblent “amochés” ; en fait c'est à cause de cet codage :

- Le fichier sera en lecture seule si vous mettez à **1** le bit **7** du premier caractère de l'extension du fichier
- Le fichier sera caché si vous mettez le bit **7** du deuxième caractère à **1**

Comme vous le voyez, il n'y a rien de bien mystérieux à ces codages ! Et si j'avais oublié quelque chose n'hésitez pas à m'en parler.

```
Sect.: 41    Taille: 00512                          Piste: 002  Nb sect.: 009

0000:  00 4D 41 58 41 4D 30 20 20 C2 C9 4E 00 00 00 34   .MAXAM0  ..N...4
0010:  02 03 04 05 06 07 08 00 00 00 00 00 00 00 00 00   ................
0020:  00 4D 41 58 41 4D 20 20 20 C2 41 53 00 00 00 0C   .MAXAM   .AS....
0030:  09 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040:  00 44 49 53 43 20 20 20 20 42 41 53 00 00 00 02   .DISC    BAS....
0050:  0B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060:  00 4D 41 58 55 53 45 52 20 A0 20 20 00 00 00 08   .MAXUSER .  ....
0070:  0C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0080:  00 4D 41 58 41 4D 31 20 20 C2 C9 4E 00 00 00 7F   .MAXAM1  ..N...▒
0090:  0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C   ................
00A0:  00 4D 41 58 41 4D 32 20 20 C2 C9 4E 00 00 00 32   .MAXAM2  ..N...2
00B0:  1D 1E 1F 20 21 22 23 00 00 00 00 00 00 00 00 00   ... !"#.........
00C0:  00 4D 41 58 41 4D 33 20 20 C2 C9 4E 00 00 00 5E   .MAXAM3  ..N...^
00D0:  24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 00 00 00 00   $%&'()*+,-./....
00E0:  00 44 49 53 4B 20 20 20 20 42 41 53 00 00 00 02   .DISK    BAS....
00F0:  30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0...............

Mode: Edition Piste     Ecriture: manuelle     Adresse:       Page: 1/2
```

Dans cet exemple d'édition d'un catalogue, on peut voir différents types de fichiers.

- MAXAMO.BIN qui est un fichier caché en lecture seule qui fait **&34** enregistrements de long
- MAXAM.BAS qui est un fichier en lecture seule de **&OC** enregistrements
- DISC.BAS qui est un fichier standard

Et vous comprenez maintenant pourquoi, du fait du système de codage des fichiers cachés et en lecture seule, certains noms de fichiers semblent amochés ! Si on ignore les bits **7**, on se retrouve l'extension du fichier telle qu'elle apparaît avec **CAT.**

LES FICHIERS

Sur CPC, nous avons deux grandes familles de fichiers : les fichiers binaires et les fichiers ASCII. La différence entre ces deux types est fondamentale.

FICHIERS BINAIRES

Les fichiers binaires sont les seuls à disposer d'un en-tête système qui permet de les identifier clairement (adresse d'implantation, taille à l'octet près, exécutable ou non, crypté ou non, etc.), ils sont forcément sauvés et chargés en mémoire en seul bloc linéaire dont la taille maximale est limitée à **64Ko.**

FICHIERS ASCII

Les fichiers dits **ASCII** sont en fait des fichiers de données brutes. Ils n'ont donc pas d'en-tête système et leur structure est gérée directement par le programme les ayant générés. Leur principal avantage par rapport aux fichiers binaires est qu'ils sont de taille quelconque et que l'on y accède sous la forme d'un flux ; il est donc possible d'y stocker des données qui ne sont pas linéaires en mémoire et de le faire en plusieurs temps.
Ainsi, tous les fichiers n'ayant pas d'en-tête système sont considérés comme étant des fichiers dits **ASCII.** En fait, seuls les fichiers dits binaires en ont en fait un.

Un fichier binaire est soit un programme **BASIC,** soit un exécutable (il contient alors directement le code machine), soit juste des données (fichiers binaires non-exécutables, comme une image sauvée avec **OCP Art Studio** par exemple).

L'EN-TÊTE SYSTÈME

L'en-tête système, quoi est-ce ? Eh bien ce sont les **128** premiers octets du premier secteur du premier bloc d'un fichier binaire. Simple non ? à quoi sert-ce ? Ce sont les quelques octets qui permettent au système de savoir qu'il s'agit effectivement d'un fichier binaire et comment il doit le traiter lors d'un **LOAD** ou d'un **RUN .**

LA STRUCTURE DE L'EN-TÊTE SYSTÈME

Dans le grand chapitre précédent, nous avons vu en détail comment était codé le catalogue et vous aurez alors sûrement remarqué qu'il lui manque pas mal de renseignements concernant les fichiers. Heureusement, l'en-tête système est là !

Cet en-tête contient donc :

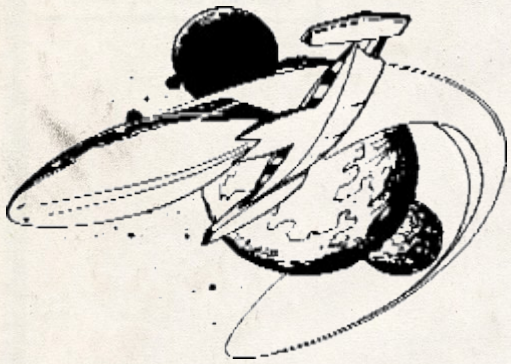
- le user et le nom de sauvegarde du fichier
- le type du fichier (normal ou protégé)
- l'adresse d'implantation du fichier
- l'adresse d'exécution du fichier
- la taille du fichier
- la somme de contrôle, le CSH (Check Sum of Header)

LE DÉTAIL

En-tête système	
Octet	**Signification**
0	« user » utilisé lors de la création du fichier (généralement à zéro).
1-8	Nom utilisé lors de la création du fichier.
9-11	Extension utilisée lors de la création du fichier.
12-17	Non utilisé.
18	Type de fichier. Deux valeurs sont possibles : 0 ou 1. La valeur 0 indique que le fichier est tout à fait normal. En revanche, s'il est à 1 cela signifie que le fichier est crypté en mode système : il s'agit d'un fichier protégé. La présence de cette valeur a deux rôles : celui d'avertir le système qu'il devra effectuer un décodage système avant de lancer le programme, mais aussi celui de provoquer l'effacement du programme s'il est interrompu durant son exécution. Pour transformer un tel fichier en fichier normal, il faudrait donc mettre cet octet à 0 puis décoder tous les secteurs un à un grâce à l'option "Coder Système" de Disco. D'une part cela impliquerait une modification du CSH (voir ci-dessous) et de plus Disco peut faire cela automatiquement avec son copieur de fichiers donc passons ; mais vous savez au moins comment ça marche !
19-20	Non utilisé.
21-22	Adresse d'implantation du fichier. Attention, c'est une valeur sur 16 bits et comme pour toute valeur gérée par l'assembleur l'octet de poids fort est après l'octet de poids faible.
23	Non utilisé.
24-25	Longueur du fichier en octet. Attention, il s'agit de la taille du fichier lui-même, sans prendre en compte les 128 octet de l'en-tête système.
26-27	Adresse d'exécution du fichier. Si l'adresse d'exécution est à **zéro** c'est généralement qu'il s'agit d'un fichier de données et non d'un programme.
28-63	Non utilisé.
64-65	Longueur du fichier en octet, oui comme pour les octets 24 et 25.
66	Non utilisé (parfois utilisé comme octet de poids fort de la taille du fichier si elle est exprimée sur 24 bits au lieu de 16).
67-68	Somme de contrôle, le CSH. Ces deux octets contiennent la somme sur 16 bits de tous les octets précédent et c'est grâce à cela que le système vérifie s'il a à faire à des données brutes (fichier ASCII) ou à un fichier binaire proprement dit. Si ces deux octets contiennent effectivement la somme de tous ceux qui précèdent, le système lit alors tous les renseignements de l'en-tête puis le lui-même fichier qui commence donc à partir du 128ème octet. Dans le cas contraire, le système considérera qu'il a à faire à un fichier ASCII et lira les données bloc par bloc à partir du tout début du secteur.
69-127	Non utilisé.

À noter que les « user », nom et extension de fichier stockés dans l'en-tête système peuvent être différents de ceux spécifié dans le catalogue si le fichier a été renommé depuis sa création.

LA FIN DES FICHIERS ASCII



Si vous avez bien suivi tout ce qui précède, vous en aurez dont subtilement déduit (Non ?! Mais si voyons) que les fichiers **ASCII** n'ont effectivement pas d'en-tête ! Mais comment qu'il fait alors le système pour savoir quand il devra arrêter de lire ? Eh bien c'est pas compliqué, ou plutôt si, c'est très compliqué ! En fait il y quasiment autant de types de détection de fin de fichiers différents que de programmes gérant des fichiers **ASCII.** Toutefois, la plupart du temps c'est par la présence d'une séquence d'octets que la fin est détectée ; c'est le cas par exemple pour les sources **Maxam** ou **Textomat.**

Il en est de même pour les fichiers **ASCII** créés depuis le **BASIC.** Dans tous les cas, la lecture d'un fichier **ASCII** s'arrêtera lorsque le nombre maximum d'enregistrements déclarés dans le catalogue aura été atteint et ce même si aucune fin de fichier n'a été détectée par l'application.

L'EXEMPLE

Voyons à présent en pratique à quoi cela ressemble avec un petit exemple à l'aide d'un grab d'un écran de Disco.

Exemple d'édition d'un en-tête système de fichier protégé :

```
Sect.: 41    Taille: 00512      Bloc: 09     Piste: 004  Nb sect.: 009

0000:  00 4D 41 58 41 4D 20 20 20 42 41 53 00 00 00 00   .MAXAM   BAS....
0010:  00 00 01 00 00 70 01 00 37 05 00 00 00 00 00 00   .....p..7.......
0020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0030:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040:  37 05 00 94 03 8B AC CD F3 9C 36 9F CC 85 0D 7A   7.........6....z
0050:  2C 3A 43 4D 44 04 60 03 81 DE E6 AD 68 1D 09 B9   ,:CMD.`.....h...
0060:  E5 BB 59 09 59 B6 E1 76 D0 13 12 F9 86 D8 9F 23   ..Y.Y..v.......#
0070:  DB 10 E9 93 26 C9 AC 42 11 CA 84 31 49 ED 2C 0D   ....&..B...1I.,.
0080:  96 2C 89 EA C1 17 30 ED 20 FF 75 0D 84 F4 4F D5   .,....0. .u...O.
0090:  24 64 23 31 FF 79 C2 60 2B 3B 81 A3 9C 43 19 55   $d#1.y.`+;...C.U
00A0:  BE ED 6E 95 12 4B F5 8A D3 D3 46 CC 09 FC 9F 31   ..n..K....F....1
00B0:  CA 71 40 88 D3 92 18 AC F0 37 C6 C1 AD CC BA 03   .q@......7......
00C0:  93 86 F4 03 00 D8 2B 5F C0 88 EE F1 AF 7B 73 B2   ......+_.....{s.
00D0:  24 C2 4A 03 0A FE 5C 35 C9 F4 A6 12 31 C5 B2 48   $.J...\5....1..H
00E0:  F4 34 00 0F 67 A7 20 84 E5 88 07 2B 4B 37 66 5B   .4..g. ....+K7f[
00F0:  8B 1A AF 5B 92 C8 39 C2 5E 2B 8A F9 2F 8D 2D 86   ...[..9.^+../.-.

Mode: Edition Fichier   Ecriture: manuelle     Adresse:       Page: 1/2
```

J'ai ici édité le fichier SYStème "MAXAM.BAS", on découvre alors grâce à l'en-tête que :

- il s'agit d'un fichier **BASIC**
- le fichier est protégé
- il s'implante en **&0170**
- il fait **&0537** octets de long
- l'adresse d'exécution n'est pas spécifiée (normal pour un fichier **BASIC**)
- le fichier est effectivement codé en mode système
  - si vous faites "**Lister -> BASIC**" : bug
  - si vous faites "**Coder -> Système**" puis "**Lister -> BASIC**" : miracle