Le M.B.R.
(Master Boot Record)

1. Structure des disques

Qu'il s'agisse de disques durs ou de disquettes, le fonctionnement est le même, la seule différence est qu'un disque dur a plusieurs partitions contrairement aux disquettes.

Un disque est constitué de plusieurs faces (on parle de têtes pour les disques durs), chaque face reçoit un certain nombre de cylindres ou pistes (suivant que l'on parle de disque dur ou de disquette), chaque cylindre est découpé en secteurs, pour les disquettes 1.44Mo il y a 18 secteurs/piste, pour les 720Ko 9 secteurs/piste et pour la majorité des disques durs modernes, 63 secteurs/cylindre (c'est le maximum que le bios supporte) et enfin, chaque secteur se compose de 512 octets.

Donc pour accéder à un secteur de 512 octets, il faut connaître :

- La face (ou tête),
- Le cylindre (ou piste)
- Le numéro du secteur

2. Structure du *MBR*

Le *MBR* se trouve sur le premier secteur (1) de la première piste (0) de la première face (0) et tient sur 1 seul secteur (512 octets).

En voici sa composition :

- Octets 0 à 1BDh : Programme d'amorce, 446 octets
- Octets 1BEh à 1CDh : 1ère partition
- Octets 1CEh à 1DDh : 2ème partition
- Octets 1DEh à 1EDh : 3ème partition
- Octets 1EEh à 1FDh : 4ème partition
- Octets 1FEh et 1FFh : Signature de secteur amorce (55AAh)

Décalage	Fonction	Taille
000h à 1BDh	Programme d'amorçage	446 octets

1BEh à 1CDh	1ère partition	16 octets
1CEh à 1DDh	2ème partition	
1DEh à 1Edh	3ème partition	
1EEh à 1FDh	4ème partition	
1FEh à 1FFh	Signature de secteur d'amorce 55AAh	2 octets

Les 446 octets du programme d'amorce vérifie la validité des données contenues dans la définition des 4 partitions, ainsi que la présence d'une partition active (et uniquement une), et une fois tout vérifié, il charge en mémoire le secteur *boot* de la partition active.

Les 4 séries de 16 octets suivantes contiennent les définitions de chaque partition.

Voici comment tout ceci est codé :

- 00h (octet) : Indicateur de partition amorçable. Les valeurs possibles sont : 0 (inactive) ou 80h (active). Si une valeur différente est présente, le programme provoquera une erreur.
- 01h (octet) : Tête ou face de début de la partition
- 02h (octet) : 8 bits de poids faible du cylindre de début (codé sur 10bits)
- 03h (octet) :
- Bits 0 à 5 = Secteur de début
- Bits 6 et 7 = Bits de poids fort du cylindre de début
- 04h (octet) : Indique le type de système installé sur la partition
- 05h (octet) : Tête ou face de fin de partition
- 06h (octet) : 8 bits de poids faible du cylindre de fin (codé sur 10bits)
- 07h (octet) :
- Bits 0 à 5 = dernier secteur
- Bits 6 et 7 = bits de poids fort du cylindre de fin
- 08h (32bits) : Adresse relative de la partition exprimée en secteurs
- 0Ch (32bits) : Nombre total de secteurs de la partition

3. Exemple d'un secteur *MBR*

```
0000:0000 eb 48 90 00 00 00 47 52 55 42 01 00 16 04 00 00 ëH....GRUB......
0000:0010 00 00 00 00 9b 1c 8f 3f b6 00 00 00 00 03 7e 78 .......?¶.....~x
0000:0020 a3 1d e0 c7 01 c8 1c e0 c7 01 01 05 1e e0 c7 01 £.àÇ.È.àÇ....àÇ.
0000:0030 dd 1c e0 c7 01 e8 00 00 58 2d 38 00 8c cf 03 02 Ý.àÇ.è..X-8..Ï..
0000:0040 80 00 00 80 a8 2c 54 01 00 08 fa 80 ca 80 ea 53 ....¨,T...ú.Ê.êS
0000:0050 7c 00 00 31 c0 8e d8 8e d0 bc 00 20 fb a0 40 7c |..1À.Ø.Ð¼. û @|
0000:0060 3c ff 74 02 88 c2 52 be 79 7d e8 34 01 f6 c2 80 <ÿt..ÂR¾y}è4.öÂ.
0000:0070 74 54 b4 41 bb aa 55 cd 13 5a 52 72 49 81 fb 55 tT´A»ªUÍ.ZRrI.ûU
0000:0080 aa 75 43 a0 41 7c 84 c0 75 05 83 e1 01 74 37 66 ªuC A|.Àu..á.t7f
0000:0090 8b 4c 10 be 05 7c c6 44 ff 01 66 8b 1e 44 7c c7 .L.¾.|ÆDÿ.f..D|Ç
0000:00a0 04 10 00 c7 44 02 01 00 66 89 5c 08 c7 44 06 00 ...ÇD...f.\.ÇD..
0000:00b0 70 66 31 c0 89 44 04 66 89 44 0c b4 42 cd 13 72 pf1À.D.f.D.´BÍ.r
0000:00c0 05 bb 00 70 eb 7d b4 08 cd 13 73 0a f6 c2 80 0f .».pë}´.Í.s.öÂ..
0000:00d0 84 f0 00 e9 8d 00 be 05 7c c6 44 ff 00 66 31 c0 .ð.é..¾.|ÆDÿ.f1À
0000:00e0 88 f0 40 66 89 44 04 31 d2 88 ca c1 e2 02 88 e8 .ð@f.D.1Ò.ÊÁâ..è
0000:00f0 88 f4 40 89 44 08 31 c0 88 d0 c0 e8 02 66 89 04 .ô@.D.1À.ÐÀè.f..
0000:0100 66 a1 44 7c 66 31 d2 66 f7 34 88 54 0a 66 31 d2 f¡D|f1Òf÷4.T.f1Ò
0000:0110 66 f7 74 04 88 54 0b 89 44 0c 3b 44 08 7d 3c 8a f÷t..T..D.;D.}<.
0000:0120 54 0d c0 e2 06 8a 4c 0a fe c1 08 d1 8a 6c 0c 5a T.Àâ..L.þÁ.Ñ.l.Z
0000:0130 8a 74 0b bb 00 70 8e c3 31 db b8 01 02 cd 13 72 .t.».p.Ã1Û¸..Í.r
0000:0140 2a 8c c3 8e 06 48 7c 60 1e b9 00 01 8e db 31 f6 *.Ã..H|`.¹...Û1ö
0000:0150 31 ff fc f3 a5 1f 61 ff 26 42 7c be 7f 7d e8 40 1ÿüó¥.aÿ&B|¾.}è@
0000:0160 00 eb 0e be 84 7d e8 38 00 eb 06 be 8e 7d e8 30 .ë.¾.}è8.ë.¾.}è0
0000:0170 00 be 93 7d e8 2a 00 eb fe 47 52 55 42 20 00 47 .¾.}è*.ëþGRUB .G
0000:0180 65 6f 6d 00 48 61 72 64 20 44 69 73 6b 00 52 65 eom.Hard Disk.Re
0000:0190 61 64 00 20 45 72 72 6f 72 00 bb 01 00 b4 0e cd ad. Error.»..´.Í
0000:01a0 10 ac 3c 00 75 f4 c3 00 00 00 00 00 00 00 00 00 .¬<.uôÃ.........
0000:01b0 00 00 00 00 00 00 00 00 c4 c5 c4 c5 00 00 80 01 ........ÄÅÄÅ....
0000:01c0 01 00 07 fe ff ff 3f 00 00 00 3b 4c 38 01 00 00 ...þÿÿ?...;L8...
0000:01d0 c1 ff 0f fe ff ff 7a 4c 38 01 86 06 70 03 00 00 Áÿ.þÿÿzL8...p...
0000:01e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000:01f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 aa ..............Uª
0000:0200
```

4. Exemple de programme

main.c

```c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/stat.h>
#include <sys/types.h>

#define TRUE  (1 == 1)
#define FALSE (1 == 0)

// table de partition
typedef struct {
    u_int8_t        etatPartition; // 1
    u_int8_t        teteLectureDebut; // 2
    u_int16_t       secteurDebut:6; // suite
    u_int16_t       cylindreDebut:10; // 3, 4
    u_int8_t        type; // 5
    u_int8_t        teteLectureFin; // 6
```

```
20      u_int16_t        secteurFin:6; // 7, 8
21      u_int16_t        cylindreFin:10; // suite
22      u_int32_t        distance; // 9, 10, 11, 12
23      u_int32_t        nombreSecteur; // 13, 14, 15, 16
24  } __attribute__((packed)) TablePartition;
25
26  // Maxi Boot Root
27  typedef struct {
28      u_int8_t        code[0x1BE]; // code d'amorçage
29      TablePartition  entreeTablePartition[4]; // table de partitions
30      u_int8_t        codeSecteurPartition[2]; // code secteur
31  } __attribute__((packed)) MBR;
32
33  // listes des types de partition
34  char *typePartition [] = {
35      "00", "01 FAT12", "02 XENIX root", "03 XENIX usr",
36      "04 FAT16 <32M", "05 Extended", "06 FAT16", "07 HPFS/NTFS",
37      "08 AIX", "09 AIX bootable", "0A OS/2 Boot Manager", "0B Win95 FAT32",
38      "0C Win95 FAT32 (LBA)", "0D", "0E Win95 FAT16 (LBA)", "0F Win95 Ext'd (LBA)",
39      "10 OPUS", "11 Hidden FAT12", "12 Compaq diagnostics",
40      "14 Hidden FAT16 <32M", "15", "16 Hidden FAT16", "17 Hidden HPFS/NTFS",
41      "18 AST SmartSleep", "19", "1A", "1B Hidden Win95 FAT32",
42      "1C Hidden Win95 FAT32", "1D", "1E Hidden Win95 FAT16", "1F",
43      "20", "21", "22", "23",
44      "24 NEC DOS", "25", "26", "27",
45      "28", "29", "2A", "2B",
46      "2C", "2D", "2E", "2F",
47      "30", "31", "32", "33",
48      "34", "35", "36", "37",
49      "38", "39 Plan 9", "3A", "3B",
50      "3C PartitionMagic recov   (FAT-12/16/32)", "3D", "3E", "3F",
51      "40 Venix 80286", "41 PPC PReP Boot", "42 SFS", "43",
52      "4C", "4D QNX4.x", "4E QNX4.x 2nd part", "4F QNX4.x 3rd part"
53      "50 OnTrack DM", "51 OnTrack DM6 Aux1", "52 CP/M", "53 OnTrack DM6 Aux3",
54      "54 OnTrackDM6", "55 EZ-Drive", "56 Golden Bow", "57",
55      "58", "59", "5A", "5B",
56      "5C Priam Edisk", "5D", "5E", "5F",
57      "60", "61 SpeedStor", "62", "63 GNU HURD or SysV",
58      "64 Novell Netware 286", "65 Novell Netware 386", "66", "67",
59      "68", "69", "6A", "6B",
60      "6C", "6D", "6E", "6F",
61      "70 DiskSecure Multi-Boot", "71", "72", "73",
62      "74", "75 PC/IX", "76", "77",
63      "78", "79", "7A", "7B",
64      "7C", "7D", "7E", "7F",
65      "80 Old Minix", "81 Minix / old Linux", "82 Linux swap", "83 Linux",
66      "84 OS/2 hidden C: drive", "85 Linux extended",
67      "86 NTFS volume set", "87 NTFS volume set",
68      "88", "89", "8A", "8B",
69      "8C", "8D", "8E Linux LVM", "8F",
70      "90", "91", "92", "93 Amoeba",
71      "94 Amoeba BBT", "95", "96", "97",
72      "98", "99", "9A", "9B",
73      "9C", "9D", "9E", "9F BSD/OS",
74      "A0 IBM Thinkpad hiberna", "A1", "A2", "A3",
75      "A4", "A5 FreeBSD", "A6 OpenBSD", "A7 NeXTSTEP",
76      "A8 Darwin UFS", "A9 NetBSD", "AA", "AB Darwin boot",
77      "AC", "AD", "AE", "AF",
78      "B0", "B1", "B2", "B3",
79      "B4", "B5", "B6", "B7 BSDI fs",
80      "B8 BSDI swap", "B9", "BA", "BB Boot Wizard hidden",
81      "BC", "BD", "BE Solaris boot", "BF",
82      "C0", "C1 DRDOS/sec (FAT-12)", "C2", "C3",
```

```
83      "C4 DRDOS/sec (FAT-16 < 32Mb)", "C5", "C6 DRDOS/sec (FAT-16)", "C7 Syrinx",
84      "C8", "C9", "CA", "CB",
85      "CC", "CD", "CE", "CF",
86      "D0", "D1", "D2", "D3",
87      "D4", "D5", "D6", "D7",
88      "D8", "D9", "DA Non-FS data", "DB CP/M / CTOS / ...",
89      "DC", "DD", "DE Dell Utility", "DF BootIt",
90      "E0", "E1 DOS access", "E2", "E3 DOS R/O",
91      "E4 SpeedStor", "E5", "E6", "E7",
92      "E8", "E9", "EA", "EB BeOS fs",
93      "EC", "ED", "EE EFI GPT", "EF EFI",
94      "F0 Linux/PA-RISC boot", "F1 SpeedStor", "F2 DOS secondary", "F3",
95      "F4 SpeedStor", "F5", "F6", "F7",
96      "F8", "F9", "FA", "FB",
97      "FC", "FD Linux raid autodetec", "FE LANstep", "FF BBT"
98  };
99
100 int main(int argc, char* argv[])
101 {
102     int f; // descripteur
103     unsigned int i; // compteur
104     MBR mbr; // structure MBR
105
106     // variable pour tester l'architecture
107     int j = 0x01020304;
108     char *p;
109
110     // ouverture du fichier associé au 1er disque dur
111     // nécessite le droit de lecture
112     if ((f = open("/dev/hda", O_RDONLY)) == -1) {
113         printf("Erreur d'ouverture du fichier /dev/hda: %s.\n",
114         strerror(errno));
115         return -1;
116     }
117
118 #ifdef DEBUG
119     // vérification de la taille des variables
120     printf("Taille d'un secteur: %d\n", sizeof(mbr));
121     printf("Taille d'une entrée: %d\n", sizeof(TablePartition));
122     printf("Taille du code: %d\n", sizeof(mbr.code));
123     printf("Taille des entrée: %d\n", sizeof(mbr.entreeTablePartition));
124     printf("Taille du code de partition: %d\n\n",
125         sizeof(mbr.codeSecteurPartition));
126     //printf("Taille du cylindre de début: %d\n\n",
127         sizeof(mbr.entreeTablePartition[0].cylindreDebut));
128 #endif
129
130     // remplir la structure mbr
131     bzero(&mbr, 512); // initialisé à zéro
132     read(f, &mbr, 512); // lire le fichier
133     close(f);
134
135     // lire les 4 entrée de la table de partition
136     for (i = 0; i < 4; i++) {
137         printf("Entrée Partition %u\n", i);
138         if (mbr.entreeTablePartition[i].type) {
139             printf("\tEtat: %0X\n", mbr.entreeTablePartition[i].etatPartition);
140             printf("\tType: %s\n",
141                 typePartition[mbr.entreeTablePartition[i].type]);
142             printf("\tTête de lecture début: %u\n",
143                 mbr.entreeTablePartition[i].teteLectureDebut);
144             printf("\tCylindre de début: %u\n",
```

```
145                 mbr.entreeTablePartition[i].cylindreDebut);
146             printf("\tSecteur de début: %u\n",
147                 mbr.entreeTablePartition[i].secteurDebut);
148             printf("\tTête de lecture fin: %u\n",
149                 mbr.entreeTablePartition[i].teteLectureFin);
150             printf("\tCylindre de fin: %u\n",
151                 mbr.entreeTablePartition[i].cylindreFin);
152             printf("\tSecteur de fin: %u\n",
153                 mbr.entreeTablePartition[i].secteurFin);
154             printf("\tDistance: %u\n", mbr.entreeTablePartition[i].distance);
155             printf("\tNombre de Secteur: %u\n",
156                 mbr.entreeTablePartition[i].nombreSecteur);
157         }
158         else {
159             printf("\tAucune entrée.\n");
160         }
161         printf("\n");
162     }
163 
164     printf("Code d'identification: %X%X.\n\n",
165         mbr.codeSecteurPartition[0],
166         mbr.codeSecteurPartition[1]);
167 
168     // tester l'architecture
169     printf("Architecture: ");
170     p = (char *) &j;    /*   int * transformé en char *   */
171     if (*p++ == 1 && *p++ == 2 && *p++ == 3 && *p++ == 4) {
172         printf("big endian.\n");
173     }
174     else {
175         p = (char *) &j;
176         if (*p++ == 4 && *p++ == 3 && *p++ == 2 && *p++ == 1)
177             printf("little endian.\n");
178             else printf("exotique !!\n");
179         }
180 
181         return EXIT_SUCCESS;
182 }
```

Résultat d'affichage

```
Taille d'un secteur: 512
Taille d'une entrée: 16
Taille du code: 446
Taille des entrée: 64
Taille du code de partition: 2

Entrée Partition 0
     Etat: 80
     Type: 07 HPFS/NTFS
     Tête de lecture début: 1
     Cylindre de début: 0
     Secteur de début: 1
     Tête de lecture fin: 254
     Cylindre de fin: 1023
     Secteur de fin: 63
     Distance: 63
```

```
     Nombre de Secteur: 20466747

Entrée Partition 1
     Etat: 0
     Type: 0F Win95 Ext'd (LBA)
     Tête de lecture début: 0
     Cylindre de début: 1023
     Secteur de début: 1
     Tête de lecture fin: 254
     Cylindre de fin: 1023
     Secteur de fin: 63
     Distance: 20466810
     Nombre de Secteur: 57673350

Entrée Partition 2
     Aucune entrée.

Entrée Partition 3
     Aucune entrée.

Code d'identification: 55AA.

Architecture: little endian.
```