CORRECTION Partiel (Sujet 1) - CYBER1 (2h00)

Architecture des Ordinateurs

NOM : PRÉNOM :

Vous devez respecter les consignes suivantes, sous peine de 0 :
— Lisez le sujet en entier avec attention
— Répondez sur le sujet
— Ne détachez pas les agrafes du sujet
— Écrivez lisiblement vos réponses (si nécessaire en majuscules)
— Les appareils électroniques sont tous interdits (calculatrices également)
— Ne trichez pas

1 Questions (10 points)

1.1 (2 points) Rappelez les 14 premières puissances de 2 :

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192

1.2 (4 points) Convertissez ces nombres en décimaux. Vous donnerez leur interprétation non-signée puis signée sur 12 bits.

	non-signé	signé
% 1101 0101 0110	3414	-682
% 1010 0001 1100	2588	-1508
$ 9BD	2493	-1603
$ A66	2662	-1434

1.3 (3 points) Convertissez ces nombres décimaux en binaire sur 12 bits, puis en hexadécimal.

	binaire	hexadécimal
42	10 10 10 0000 0010 1010	2A 02A
2341	1001 0010 0101	925
-116	1111 1000 1100	F8C

1.4 (1 point) Convertissez ces nombres en flottants au format IEEE 754 simple précision :

	exposant									hexadécimal								
$68,78125$	%	1	0	0	0	0	1	0	1	$	4	2	8	9	9	0	0	0

2 Problème (10 points)

Afin de vous plonger réellement dans le *forensic* (analyse forensique ou investigation numérique en français), une toute petite FAT12 a été réalisée et utilisée pour produire quelques fichiers et dossiers contenant quelques mots. L'objectif est de retrouver ce contenu.

Cet exercice a été réalisé avec une série de commandes que vous pourrez tester de votre côté pour également observer l'évolution d'une petite FAT12 (voir **`truncate(1)`**, **`mkfs(1)`** et **`mount(1)`**). Pour continuer l'examen, vous pouvez aller directement à la partie suivante : *Description du système de fichiers FAT*.

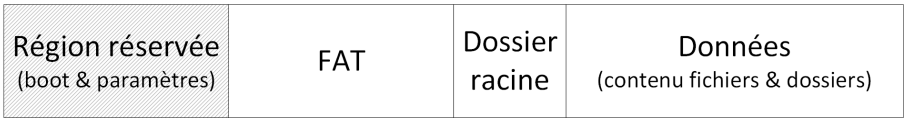
Les commandes ayant permis de produire et utiliser la FAT sont les suivantes :

```
touch Disk.img
truncate Disk.img -s 50K
# FAT12, secteurs de 512o, 1 secteur par cluster
mkfs.vfat -F12 -S512 -s1 Disk.img
sudo mount Disk.img /mnt
### ... ###
cd /
sudo umount /mnt
```

Cependant, la plus petite FAT possible avec les outils linux reste une FAT12 de 50 kilo-octets, c'est-à-dire, toujours trop pour tenir sur une ou deux feuilles A4 en examen. N'oubliez pas d'utiliser *ghex* (ou d'autres éditeurs hexadécimaux) régulièrement pour observer les changements d'état de la FAT lorsque vous expérimenterez chez vous.

Description du système de fichiers FAT

Le format FAT (*File Allocation Table*) est relativement simple, mais de nombreux champs dans les structures ainsi que des régions ne seront pas utiles dans notre cas. Pour résumer, une partition formatée en FAT se divise en 4 régions, mais nous nous concentrerons principalement sur les régions du dossier racine et celle des données.



Pour identifier des fichiers et des dossiers parmi les données, il est nécessaire d'étudier les *directory entries* (ou *direntry*) : des structures de données contenant les caractéristiques des objets stockés dans la partition.

Un dossier est littéralement un tableau contant des structures de données décrivant chacune un fichier ou un dossier. Il y a donc autant de cases dans le tableau qu'il y a de fichiers et dossiers contenus à ce niveau hiérarchique.

`dossier/`
`dossier/fichier1`
`dossier/fichier2`
`dossier/fichier3`

```
struct direntry   entries[3]; // dossier
entries[0];       // dossier/fichier1
entries[1];       // dossier/fichier2
entries[2];       // dossier/fichier3
```

La structure représentant une *direntry* est de la forme suivante. On notera que FAT12 est très limité, ainsi, les fichiers ont des noms de 11 caractères maximum (8 avant l'extension, et 3 après). À noter : la taille enregistrée dans le champs *size* est en octets.

```
struct direntry {
  char[11] name;
  char     attributes;
  char[14] reserved_and_dates;
  int      first_cluster;
  long     size;
} __attribute__((packed))
```

Les types de données font :

char	:	1 octet	(8 bits)
int	:	2 octets	(16 bits)
long	:	4 octets	(32 bits)

Attributs :

ATTR_READ_ONLY	0x01
ATTR_HIDDEN	0x02
ATTR_SYSTEM	0x04
ATTR_VOLUME_ID	0x08
ATTR_DIRECTORY	0x10
ATTR_ARCHIVE	0x20

Taille en octets d'une *direntry* : 32 octets *(0,5 pts)*

Pour vous aider à retrouver les chaînes de caractères, une table ASCII décimale/caractères est fournie :

Dec	10	13	32	45	46		48	49	50	51	52	53	54	55	56	57
Char	\n	\r	*(espace)*	-	.		0	1	2	3	4	5	6	7	8	9

Dec	65	66	67	68	69	70	71	72	73	74	75	76	77
Char	A	B	C	D	E	F	G	H	I	J	K	L	M

Dec	78	79	80	81	82	83	84	85	86	87	88	89	90
Char	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

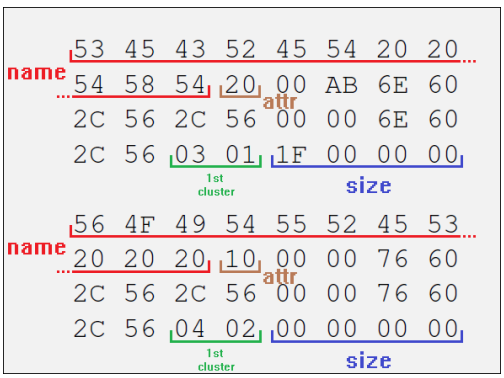
Dec	97	98	99	100	101	102	103	104	105	106	107	108	109
Char	a	b	c	d	e	f	g	h	i	j	k	l	m

Dec	110	111	112	113	114	115	116	117	118	119	120	121	122
Char	n	o	p	q	r	s	t	u	v	w	x	y	z

2.1 (2 points) Première étape : séparation des champs du dossier racine

Le dossier racine (ou *root directory* en anglais) est le premier dossier dans lequel des fichiers et des dossiers peuvent se trouver. La région du dossier racine contient en réalité une *direntry* avec quelques valeurs spécifiques.

En lisant la région du dossier racine, on extrait les données suivantes. Recopiez les différents champs dans le tableau associé, sans les interpréter pour le moment.



	direntry[0] (f1)						direntry[1] (f2)					
name	53	45	43	52	45	54	56	4F	49	54	55	52
	20	20	54	58	54		45	53	20	20	20	
attributes	20						10					
first_cluster	03 01						04 02					
size	1F 00 00 00						00 00 00 00					

2.2 (2 points) Deuxième étape : conversion des champs

Maintenant que vous avez extrait les champs, il est nécessaire de les convertir pour obtenir des valeurs interprétables. Convertissez de l'hexadécimal vers le décimal pour retrouver les caractères.

Concernant les noms de fichiers, les normes FAT12 et FAT16 précisent que les 8 premiers caractères servent à coder le nom du fichier, et les 3 derniers servent à coder l'extension. Si un caractère correspond à un espace, laissez sa case vide.

direntry[0] (f1)	S	E	C	R	E	T			.	T	X	T
direntry[1] (f2)	V	O	I	T	U	R	E	S	.			

Retrouvez maintenant les attributs de chaque direntry, puis convertissez les entiers en valeurs décimales.

Avant de convertir les entiers, il faut savoir qu'en FAT, les entiers sont codés en *little endian* (*petit boutiste* en français), c'est-à-dire que les octets sont écrits au fur et à mesure du plus petit poids au plus grand. Ainsi, si on écrivait « 1234 » en little endian par paquets de un chiffre, on écrirait « 4321 », car 4 a le poids le plus petit (celui des unités), et 1 a le poids le plus fort (celui des milliers). Pour effectuer les conversions d'entiers, vous devrez donc inverser l'ordre de lecture des octets avant de les convertir (ainsi, « BE 3F » doit être interprété comme « 3F BE » avant d'être converti en décimal, « AB CD EF » doit être interprété comme « EF CD AB », et ainsi de suite).

	taille (en octets)	numéro du premier cluster	attributs
direntry[0] (f1)	1F 00 00 00 → 00 00 00 1F 31	03 01 ⟶ 01 03 259	☐ Read Only ☐ Hidden ☐ Volume ID ☐ System ☐ Directory X Archive
direntry[1] (f2)	00 00 00 00 → 00 00 00 00 0	04 02 ⟶ 02 04 516	☐ Read Only ☐ Hidden ☐ Volume ID ☐ System X Directory ☐ Archive

2.3 (1 point) Troisième étape : lecture d'un fichier

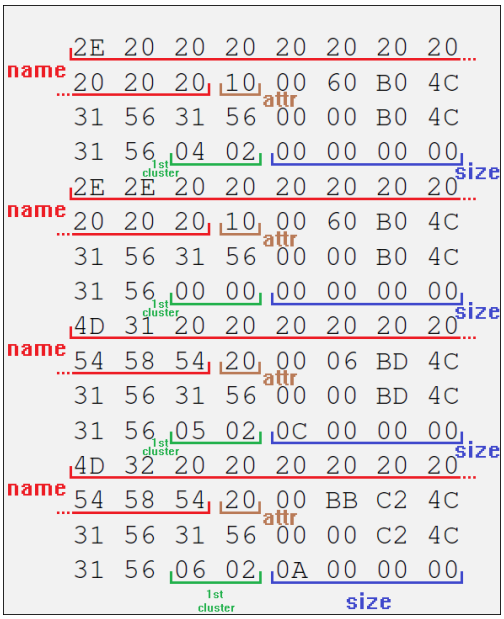
L'une des direntry précédente a un nom particulièrement intéressant, et il est clair qu'une information importante se trouve dans le cluster pointé. Voici les données extraites du cluster dont il est question. Convertissez le message contenu dans le fichier, mais n'oubliez pas de vous arrêtez à la taille en octets indiquée par la direntry associée (c'est-à-dire f1). Faites attention à la casse, c'est-à-dire aux majuscules et minuscules lorsque vous écrirez votre réponse.

```
4C 65 20 73 65 63 72 65
74 20 61 20 63 68 65 72
63 68 65 72 20 65 73 74
20 45 50 49 54 41 0A 54
4F 50 20 65 63 6F 6C 65
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

L	e		s	e	c	r	e
t		a		c	h	e	r
c	h	e	r		e	s	t
	E	P	I	T	A	\n	T
O	P		e	c	o	l	e

2.4 (3 points) C'est reparti...

Le message récupéré semble parfaitement clair, mais c'est trop simple pour être la réponse attendue. La deuxième direntry (f2) dispose peut être de la réponse. Le cluster pointé par celle-ci renvoie ces données, remplissez les structures qui devraient logiquement lui être adjointes.



	direntry[0] (f11)						direntry[1] (f12)					
name	2E	20	20	20	20	20	2E	2E	20	20	20	20
	20	20	20	20	20		20	20	20	20	20	
attributes	10						10					
first_cluster	04 02						00 00					
size	00 00 00 00						00 00 00 00					

	direntry[2] (f13)						direntry[3] (f14)					
name	4D	31	20	20	20	20	4D	32	20	20	20	20
	20	20	54	58	54		20	20	54	58	54	
attributes	20						20					
first_cluster	05 02						06 02					
size	0C 00 00 00						0A 00 00 00					

direntry[0] (f11)	.								.			
direntry[1] (f12)	.	.							.			
direntry[2] (f13)	M	1							.	T	X	T
direntry[3] (f14)	M	2							.	T	X	T

	taille (en octets)	numéro du premier cluster	attributs
direntry[0] (f11)	00 00 00 00 → 00 00 00 00 0	04 02 ⟶ 02 04 516	☐ Read Only ☐ Hidden ☐ Volume ID ☐ System X Directory ☐ Archive
direntry[1] (f12)	00 00 00 00 → 00 00 00 00 0	00 00 ⟶ 00 00 0	☐ Read Only ☐ Hidden ☐ Volume ID ☐ System X Directory ☐ Archive
direntry[2] (f13)	0C 00 00 00 → 00 00 00 0C 12	05 02 ⟶ 02 05 517	☐ Read Only ☐ Hidden ☐ Volume ID ☐ System ☐ Directory X Archive
direntry[3] (f14)	0A 00 00 00 → 00 00 00 0A 10	06 02 ⟶ 02 06 518	☐ Read Only ☐ Hidden ☐ Volume ID ☐ System ☐ Directory X Archive

2.5 (1 point) Système, dossiers, clusters, et pointeurs

Avec toutes les méta-données réunies jusqu'à maintenant concernant f1, f2, f11, f12, f13, f14, que déduisez-vous à propos de f11 et f12 ? (observez particulièrement les numéros de clusters)

2.6 (1 point) C'est reparti... [bis]

Finalement, il reste encore deux derniers clusters à convertir. Attention à la taille des données, ainsi qu'aux majuscules et minuscules.

```
56 65 67 61 2D 4D 69 73
73 79 6C 0A 6E 31 0A 00
```

V	e	g	a	-	M	i	s
s	y	l	\n	n	1	\n	

```
43 68 6F 75 70 65 74 74
65 0A 00 00 00 00 00 00
```

C	h	o	u	p	e	t	t
e	\n						

SUJET 1