Stockage des données – Corrigés

Exercice 1. La seconde loi de Moore postule que la capacité des microprocesseurs (en équivalents de nombres de transistors) double tous les 2 ans. Il s'agit donc d'une croissance exponentielle.

Exercice 2.

1. Comme 400 Kio correspond à $400 \times 2^{10} = 409\,600$ octets, la capacité en ko des premières disquettes de Sony était d'environ 410 ko.
2. Comme 76,688 Gio correspond à $76{,}688 \times 2^{30} \approx 82{,}3 \times 10^9$ octets, un disque dur S-ATA Hitachi de fin 2005 avait une capacité de stockage d'environ 82,3 Go.

Exercice 3.

1. Le texte compte 104 lettres, 4 caractères de ponctuation, 23 espaces et 2 retours à la ligne donc la taille du fichier texte codé en ASCII est $104 + 4 + 23 + 2 = 133$ octets.
   (Dans la pratique, on obtient un fichier de 135 octets car il semble que les caractères accentués soient codés sur 2 octets et non pas 1 octet comme les autres caractères.)
2. Le fichier compte $12 \times 30 = 360$ caractères ainsi que 11 retours à la ligne donc sa taille est $360 + 12 = 372$ octets.
3. Chaque caractère occupant 1 octet, un fichier de 236 ko contient au maximum 236 caractères.

Exercice 4. On peut classer les fichiers de la façon suivante :

- fichiers texte : `fichier2.txt` et `fichier7.doc`
- fichiers image : `fichier1.jpg` et `fichier5.png`
- fichiers son : `fichier9.wav` et `fichier10.mp3`
- fichiers vidéo : `fichier3.mov`, `fichier6.mp4` et `fichier8.avi`
- fichier exécutable : `fichier4.exe`

Exercice 5. En se référant aux ordres de grandeurs standards, `fichier1` est une vidéo donc son extension est .avi, `fichier2` est un fichier son donc son extension est .wav et `fichier3` est un fichier texte donc son extension est .txt.

Exercice 6.

1. L'image contient $240 \times 240 = 57\,600$ pixels donc, comme chaque pixel est codé sur un octet, la taille de l'image est 57,6 ko.
2. **a.** En ne conservant qu'une ligne sur deux et qu'une colonne sur deux, on obtient une image comptant $120 \times 120 = 14\,400$ pixels et sa taille est donc 14,4 ko.
   b. En ne conservant que 16 niveaux de gris, on peut coder chaque niveau de gris sur 3 bits (car $2^3 = 8$) et donc chaque pixel nécessite 3 bits de mémoire. Ainsi, la taille de l'image est $57\,600 \times 3 = 172\,000$ bits c'est-à-dire $\dfrac{172\,000}{8} = 21\,600$ octets soit finalement 21,6 ko.

Exercice 7.

1.

```
def en_binaire(N):
    O=' '
    for i in range(8):
        b=str(N//2**(7-i))
        O=O+b
        N=N%2**(7-i)
    return(O)
```

2.

```
def en_decimal(O):
    N=0
    for i in range(8):
        N=N+int(O[7-i])*2**i
    return(N)
```

3. On utilise la table ASCII suivante :

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}

Ainsi, `Binaire` se traduit en décimal 66 105 110 97 105 114 101 et, grâce à la fonction `en_binaire`, on en déduit que le code ASCII du mot `Binaire` est

`01000010 01101001 01101110 01100001 01101001 01110010 01100101`

4. Grâce à la fonction `en_decimal`, on trouve que la suite d'octets

`01000100 01100101 01100011 01101001 01101101 01100001 01101100`

correspond à la suite d'entiers 68 101 99 105 109 97 108 et donc, en utilisant la table précédente, on conclut que le mot correspondant est `Decimal`.