Bonjour tout le monde

Aujourd'hui je vais vous apprendre comment un ordinateur fait pour afficher un texte.
Pour ceux qui pensaient que c'était facile, désolé, cela ne l'est pas.

Déjà un ordinateur possède plusieurs composants :

- La RAM qui stocke des octets. Et qui a un temps de latence d'un millionième de seconde.
  (La RAM « mémoire vive » stocke les octets que lorsque l'ordinateur est sous tension.)

- Le disque dur qui stocke aussi des octets. Qui a une latence d'un millième de seconde
  (Le disque dur « mémoire morte » stocke les octets même lorsque l'ordinateur est éteint)
  (c'est pour cela que la RAM existe, elle va mille fois plus vite et le disque dur permet de stocker vos donnés, exemple : les fichiers)

Les octets : Un octet est une suite de 8 0 et 1. exemple : 0110 0101
Où 0 est absence de courant et 1 présence de courant.
Cela donne en base 10 un nombre entre 0 et 255 (soit 256 possibilités)
Dans la mémoire de l'ordinateur, rien n'indique à quoi correspond un octet stocké.
Du code, du texte, des images, des vidéos, du sons, de la 3d … etc
L'ordinateur va être amené à traiter ces donnés de la bonne manière.

- Le processeur exécute des instructions présentes dans la RAM et possède un petit nombre de registres.
  Un registre stocke un octet et l'association de plusieurs registres ensemble permet de gérer des nombres plus grand.
  (le temps de latence des registres est d'un milliardième de seconde , c'est donc la mémoire la plus rapide mais l'ordinateur ne possède environ qu'une dizaine de registres.)

Dans la RAM, les octets se suivent avec un numéro d'index : le premier , le deuxième , le troisième … etc
Le processeur possède un registre d'exécution de grande taille dont la valeur correspond à un index dans la RAM.
Le processeur exécute cette instruction dans la RAM pointé par son registre d'exécution puis passe à la suivante (et cela va très vite, par exemple, pour un ordinateur de 2 giga hertz, celui ci possède 2 milliard de cycle processeur par secondes. Les instructions prennent 1 ou peu de cycle pour être exécuté.)

Les instructions du processeur, pour faire tout ce qu'on veut, sont stocké dans la RAM et chaque nombre entre 0 et 255 correspond à une instruction.

1. Celles pour gérer les registres entre eux (addition, soustraction et d'autre plus compliquées)
2. Celles pour accéder à la RAM (un registre pointe un index de la RAM et une autre instruction pour récupérer ou changer cette valeur pointé)
3. Des conditions (des calcul conditionnel sur les registres)
4. Des saut (le registre d'exécution saute à un autre index spécifié)
5. Des sauvegarde de registre dans la pile (stocke les registres pour les récupérer plus tard, *je n'explique pas son fonctionnement exacte dans ce cour.* Sachez que cela est utilisé quand on appel des sous scripts)
6. Des instructions pour gérer les périphériques (disque dur, clavier, souris, écran, carte graphique, port USB, son … etc)

Le fonctionnement général du processeur est simple :
Il suit des instructions dans la RAM qui lui fait gérer ses registres avec lesquels il gère la RAM.

- L'écran qui permet d'afficher des images et du texte a un fonctionnement simple :
  Le processeur lui envoi les trois couleurs primaire des écrans (le rouge , le vert , le bleu)
  d'un octet par couleur et cela pour tout les pixels de l'écran.

Lorsque l'on lance le système d'exploitation ou le programme qui va écrire notre texte, un petit script d'instructions processeur va copier du disque dur dans la RAM le code et les donnés du programme, c'est toujours de cette manière quel que soit les donnés en question car la RAM est plus rapide.

Donc dans notre cas, pour écrire du texte, notre programme possède la phrase en question à écrire. Cette phrase est codé dans la RAM selon la correspondance ASCII où chaque valeur d'un octet correspond à un caractère :

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

Le script de notre programme va donc récupérer successivement les caractères de notre phrase à afficher, un script fait pour afficher le caractère va être appelé. Le résultat est que le caractère sera écrit en pixel dans une zone de mémoire de la RAM qui stocke les couleurs de l'écran.
(Aujourd'hui il existe beaucoup plus de caractères, c'est pourquoi plusieurs octets sont utilisés pour coder chaque caractère)

En effet chaque couleur : rouge,vert,bleu ,d'un octet respectivement, de toutes les pixels de votre écran sont stocké dans une grande zone mémoire dans votre RAM.
Ce n'est qu'une fois le travaille graphique achevé que l'ordinateur exécute un script qui envoie à

l'écran les valeurs de couleurs de chaque pixels.

Voilà, dans ce cour je vous épargne le code des scripts utilisé mais j'imagine que vous arrivé à vous représenter leur complexité. Donc non, écrire un caractère, ce n'est pas simple.

A bientôt