Programmation C et C++

Création, chargement, sauvegarde et traitement d'images tiff, rgba, non compressées

C.Turrier – 20 novembre 2020

1) Types des données, en C et en C++

Le programme suivant montre les principaux types de données et plages correspondantes sous Ubuntu 18.04 64bits.
On voit que pour décrire les données images et leurs adresses, on peut utiliser le type **unsigned int**.

```
//-----------------------------------------------
//test.cpp
//Compilation en C++: g++ -Wall test.cpp -lm -o test
//-----------------------------------------------
#include <stdio.h>
int main(int argc, char **argv)
{
unsigned int x;
printf("sizeof(short): %d\n", (int) sizeof(short));
printf("sizeof(int): %d\n", (int) sizeof(int));
printf("sizeof(unsigned int): %u\n", (int) sizeof(unsigned int));
printf("sizeof(long): %d\n", (int) sizeof(long));
printf("sizeof(long long): %d\n", (int) sizeof(long long));
printf("sizeof(size_t): %d\n", (int) sizeof(size_t));
printf("sizeof(void *): %d\n", (int) sizeof(void *));
x=255 + 256*255 + 65536*255 + 16777216*255U;
printf("x= %u\n", x);
printf("Saisir un chiffre puis la touche Entrée, pour quitter !\n");
scanf("%u", &x);
return 0;
}
ubuntu@ubuntu-MS-7C08:/media/ubuntu/data/C++CLAUDE$ ./test
sizeof(short): 2
sizeof(int): 4
sizeof(unsigned int): 4
sizeof(long): 8
sizeof(long long): 8
sizeof(size_t): 8
sizeof(void *): 8
x= 4294967295
x= 4294967295
```

2) Arrangement des octets

D’une façon générale, les données (nombres ou textes) sont représentées, en mémoire ou dans un fichier, par des octets. Chaque octet contient 8 éléments binaires 0 ou 1.

Exemple :
Un nombre n de type unsigned int est réprésenté par 4 octets. Il varie donc dans la plage suivante :
[0, +4294967295] lorsque ce nombre n est affiché en décimal
[00 00 00 00, FF FF FF FF] lorsque ce nombre n est affiché en hexadécimal.

Les données sont représentées selon un arrangement des octets qui peut être de type «big endian» ou de type «little endian»

Avec un arrangement des octets de type «big endian», les octets de poids fort sont placés en premier.
En revanche, avec un arrangement des octets de type « little endian», *ce sont* les octets de poids faibles qui sont placés en premier.

Ainsi, soient 4 octets o1, o2, o3, o4 placés dans l’ordre indiqué
En représentation «little endian», le nombre unsigned int n associé à ces 4 octets est :
n = o1 + 256*o2 + 256*256*o3 + 256*256*256*o4
n= o1 + 256*o2 + 65.536*o3 + 16.777.216*o4
En représentation «big endian», le nombre unsigned int n associé à ces 4 octets est :
n = 256*256*256*o1 + 256*256*o2 o2 + 256*o3 + o4
n= 16.777.216*o1 + 65.536*o2 + 256*o3 + o4

Exemple:
Le nombre n=242 en décimal équvaut à F2 00 00 00 en représentation «little endian», et à 00 00 00 F2 en représentation «big endian»

2) Structure d’un fichier tiff

Dans ce qui suit on utilise une présentation des octets de type little endian (format adopté par gimp par défaut).
La structure minimale d’un fichier tiff contenant une image rgba, non compressée dont les pixels sont organisés en une seule bande est la suivante :

Header	**Pixels**	**IFD**															**Divers**	
H	P	N	IFE1	IFE2	IFE3	IFE4	IFE5	IFE6	IFE7	IFE8	IFE9	IFE10	IFE11	IFE12	IFE13	F	R	Q
8	4*w*h	2	12	12	12	12	12	12	12	12	12	12	12	12	12	4	16	8

(H)eader entête : 8 octets
(P)ixels rbga : 4*w*h
----------début de l’IFD (Image File Directory)----------
(N)ombre d'IFEs (Image Files Entries) 2 octets
IFE1 à 13 : 13x12 octets =156
(F)in d'IFD : 4 octets
----------fin de l’IFD----------
(R)ésolution h et v: 2x8 =16 octets
(Q)uantité de bits pour chaque composantes rgba : 4x2=8 octets

D’où le nombre total d'octets d’un fichier tiff rgba, non compressé= 194 + 4*w*h

2.1) Entête (8octets)

49	49	2A	00	o1	o2	o3	O4
little endian		tiff		adresse du 1er octet de l’IFD dans le fichier			

2.2) Pixels rgba (4*w*h octets)

w : largeur de l’image en pixels
h : hauteur de l’image en pixels

rr	bb	gg	aa	rr	bb	gg	aa	rr	bb	gg	aa	rr	bb	gg	aa
pixel 1				pixel 2				pixel 3				pixel 4			

...

rr	bb	gg	aa	rr	bb	gg	aa	rr	bb	gg	aa	rr	bb	gg	aa
pixel 4*w*h-3				pixel 4*w*h-2				pixel 4*w*h-1				pixel 4*w*h			

Les composantes de couleurs rr, bb, gg et aa de chaque pixel peuvent varier de 00 à FF en hexadécimal, c’est à dire 0 à 255 en décimal.

2.3) IFD (162 octets)

L'IFD comprend les 13 IFEs obligatoires suivants (pour une image rgba non compressée), placés en ordre croissant t précédés du nombre 13 (nombre d'iFEs) codé sur 2 octets. L'IFD se termine par 4 octets 00.

2.3.1) Nombre d'IFEs (2 octets)

0C	00

= 13 en décimal = nombred'IFEs

2.3.2) IFEs (156 octets)

Les 13 IFEs suivants de 12 octets chacun sont obligatoires pour les images tiff RVBA (13x12= 156 octets)

TagName	**Tag**	**Tag**	**Par exemple**
IFE1: ImageWidth	256	0001	00, 01 | 03, 00 | 01, 00, 00, 00 | **58, 02**, 00, 00 (ici, 600px en decimal)
IFE2: ImageLength	257	0101	01, 01 | 03, 00 | 01, 00, 00, 00 | **90, 01**, 00, 00 (ici 400px en décimal)
IFE3: BitsPerSample	258	0201	02, 01 | 03, 00 | 04, 00, 00, 00 | **F6, A6, 0E, 00** (à cette adresse :8,8,8,8)
IFE4: Compression	259	0301	03, 01 | 03, 00 | 01, 00, 00, 00 | **01**, 00, 00, 00 (=1=> sans c)
IFE5: PhotometricInterpret.	262	0601	06, 01 | 03, 00 | 01, 00, 00, 00 | **02**, 00, 00, 00 (=2=>rgb)
IFE6: StripOffsets (nb bandes)	273	1101	11, 01 | 04, 00 | **01**, 00, 00, 00 | **08**, 00, 00, 00 (=1 seule bande à l'adr 8)
IFE7: SamplesPerPixel	277	1501	15, 01 | 03, 00 | 01, 00, 00, 00 | **04**, 00, 00, 00 (=4=>rgba)
IFE8: RowsPerStrip	278	1601	16, 01 | 03, 00 | 01, 00, 00, 00 | **xx**, **yy**, 00, 00 (xx, yy= h pixels)
IFE9: StripByteCounts	279	1701	17, 01 | 04, 00 | 01, 00, 00, 00 | **aa**, **bb**, 00, 00 (aa,bb =w*h*4)
IFE10: XResolution	282	1A01	1A, 01 | 05, 00 | 01, 00, 00, 00 | **E6, A6, 0E, 00** (à cette adr : 8octets resX)
IFE11: YResolution	283	1B01	1B, 01 | 05, 00 | 01, 00, 00, 00 | **EE, A6, 0E, 00** (à cette adr : 8octets resY)
IFE12: ResolutionUnit	296	2801	28, 01 | 03, 00 | 01, 00, 00, 00 | **02**, 00, 00, 00 (2=>inch)
IFE13: ExtraSample (canal a)	338	5201	52, 01 | 03, 00 | 01, 00, 00, 00 | **02**, 00 00 00 (2=>canal alpha existe)

2.3.3) Fin de l’IFD (4 octets)

0	00	00	00

2.4) Divers

2.4.1) Résolution horizontale et verticale de l’image (16 octets)

48	00	00	00	01	00	00	00	48	00	00	00	01	00	00	00

(48, 00, 00, 00) = 72 en décimal et (00, 00, 00, 01) signifie 72/1 = 72 dpi (résolution de l'image)

(Q)uantité de bits pour chaque composantes rgba : 4x2=8 octets

2.4.2) Quantité de bits pour chaque composantes rgba : (8 octets)

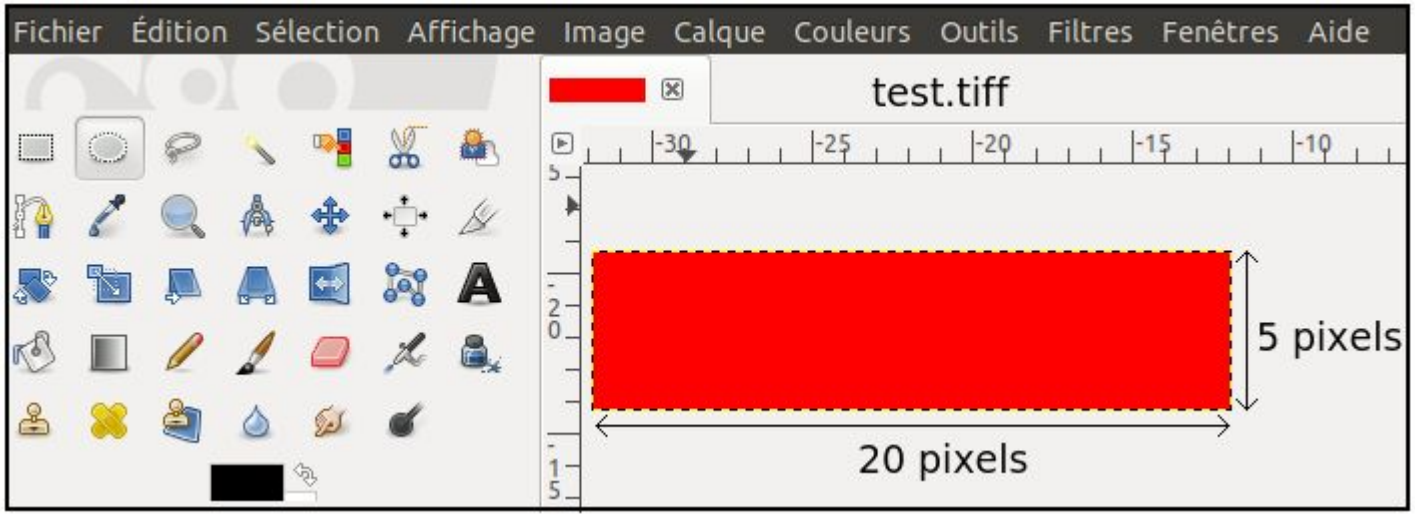
08	00	08	00	08	00	08	00
8 bits pour r		8 bits pour g		8 bits pour b		8 bits pour a	

(08, 00) signifie 8 bits pour la composante r, g, b et a

2.5) Exemple : fichier test.tiff

On examine le contenu du fichier tiff contenant une image de 20x5 pixels, rgba, non compressée dont les pixels sont organisés en une seule bande et dont les octets sont arrangés selon le type little endian
On voit que les 100 pixels sont codés FF,00,00,FF, ce qui correspond à une couleur rouge entièrement opaque.
On retrouve la structure détaillée ci-dessus avec notamment les 13 IFEs, coloriés ici en orange et vert.

```
Image test.tiff largeur 20 pixels, hauteur 5 pixels
00000000 49 49 2A 00 98 01 00 00 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000039 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000078 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF
00000117 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF
00000156 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000195 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000234 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF
00000273 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF
00000312 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000351 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000390 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF 0D 00 00 01 03 00 01 00 00 00 14 00 00 00 01 01 03 00 01 00 00
00000429 00 05 00 00 00 02 01 03 00 04 00 00 00 4A 02 00 00 03 01 03 00 01 00 00 00 01 00 00 00 06 01 03 00 01 00 00 00 02 00
00000468 00 00 11 01 04 00 01 00 00 00 08 00 00 00 15 01 03 00 01 00 00 00 04 00 00 00 16 01 03 00 01 00 00 00 05 00 00 00 17
00000507 01 04 00 01 00 00 00 90 01 00 00 1A 01 05 00 01 00 00 00 3A 02 00 00 1B 01 05 00 01 00 00 00 42 02 00 00 28 01 03 00
00000546 01 00 00 00 02 00 00 00 52 01 03 00 01 00 00 00 02 00 00 00 00 00 00 00 48 00 00 00 01 00 00 00 48 00 00 00 01 00 00
00000585 00 08 00 08 00 08 00 08 00
```



3) Fonctions C

Les fonctions suivantes sont écrites en langage C et sont compatibles C++. Elles supportent uniquement les images tiff, rgba, non compressées, avec les octets arrangés selon le type little endian dans les fichiers tiff.

unsigned char * tif_load_file_to_table (**char** * filename, **int** info);
void tif_save_table_to_tiff_file (**unsigned char** * table, **char** * filenamedest, **int** info);

unsigned char* tif_create_table (**unsigned int** width, **unsigned int** height, **int** info);
unsigned char * tif_load_file_to_table (**char** * filename, **int** info);
void tif_save_table_to_data_file (unsigned **char** * t able, **char** * filenamedest, **int** info);
void tif_save_table_to_tiff_file (**unsigned char** * table, **char** * filenamedest, **int** info);
void tif_table_set_pixel (**unsigned char** * table, **int** x, **int** y, **int** r, **int** g, **int** b, **int** a);
void tif_table_draw_rectangle (**unsigned char** * table, **int** x0, **int** y0, **int** x1, **int** y1, **int** r, **int** g, **int** b, **int** a);
unsigned char * tif_decimal_to_4bytes (**unsigned int** n, **int** info);
unsigned int tif_4bytes_to_decimal (**int** o1, **int** o2, **int** o3, **int** o4);

La fonction **tif_create_table()** permet de créer une table t en mémoire crrespondant à une image rgba blanche opaque, de largeur w et de hauteur h.
A l'issue de l'opération, la table t contient dans l'ordre, la largeur w et la hauteur h de l'image (sur 2 octets chacun) suivis de l'ensemble des w*h*4 pixels rgba, soit un total de 4+4*w*h octets
Le paramètre info (valeur 1 ou 0) permet d'afficher ou non un compte rendu, à l'issue de l'opération

La fonction **tif_load_file_to_table()** permet de charger le contenu d'un fichier tiff dans une table t en mémoire.
A l'issue de l'opération, la table t contient dans l'ordre, la largeur w et la hauteur h de l'image (sur 2 octets chacun) suivis de l'ensemble des w*h*4 pixels rgba, soit un total de 4+4*w*h octets
Le paramètre info (valeur 1 ou 0) permet d'afficher ou non un compte rendu, à l'issue de l'opération. Cette fonction écrit dans la table uniquement les valeurs de la largeur, de la hauteur et des pixels de l’image.

Elle ne conserve pas les autres valeurs (IFD, IFE…) qui se trouvaient dans le fichier tiff.

Exemple :
unsigned char *t=NULL;
t=tif_load_file_to_table("essai01.tiff",1);

La fonction **tif_save_table_to_data_file()** permet d'enregistrer le contenu d'une table image mémoire t à l'identique, dans un fichier de données brutes.
Exemple :
unsigned char *t=NULL;
tif_save_table_to_data_file (t, "essai.dat", 0);

La fonction **tif_save_table_to_tiff_file()** permet d'enregistrer le contenu d'une table image mémoire t dans un fichier image au format tiff de type rgba, non compressé, en format little endian.
En plus de l'enregistrement des 4*w*h pixels rgba, la fonction reconstruit l'ensemble des octets nécessaires et notamment l'en-tête (8 octets), le nombre d'IFEs (2octets) , les 13 IFEs (156+4 octets) qui sont obligatoires pour décrire le type rgba, non compressé ainsi que 24 les octets complémentaires en fin de fichier.
Exemple :
unsigned char *t=NULL;
tif_save_table_to_tiff_file (t, "essai.tiff", 0);

La fonction **tif_table_set_pixel()** pemet de modifier, dans une table t en mémoire, la valeur des composantes de couleur rgb et a d'un pixel situé à l'abcisse x et à l'ordonnée y dans cette table.

La fonction **tif_table_draw_rectangle()** permet de dessiner, dans une table t en mémoire, un rectangle (x0,y0) à (x1,y1) de couleur r,g,b,a.

Les fonctions **tif_decimal_to_4bytes()** et **tif_4bytes_to_decimal()** permettent de convertir un nombre de 4 octets en un entier et inversement.

Résumé

Les octets image, dans un fichier tiff, dans une table t en mémoire et dans un fichier data se répartissent de la façon suivante.

Dans un fichier tiff, on trouve, en plus des valeurs des composantes de couleur r,g,b et a des pixels, les octets relatifs à l'en-tête (Header), à l'Image File Directory (IFD) , à la (R)ésolution de l'image et à la (Q)uantité de bits pour chaque composantes rgba.

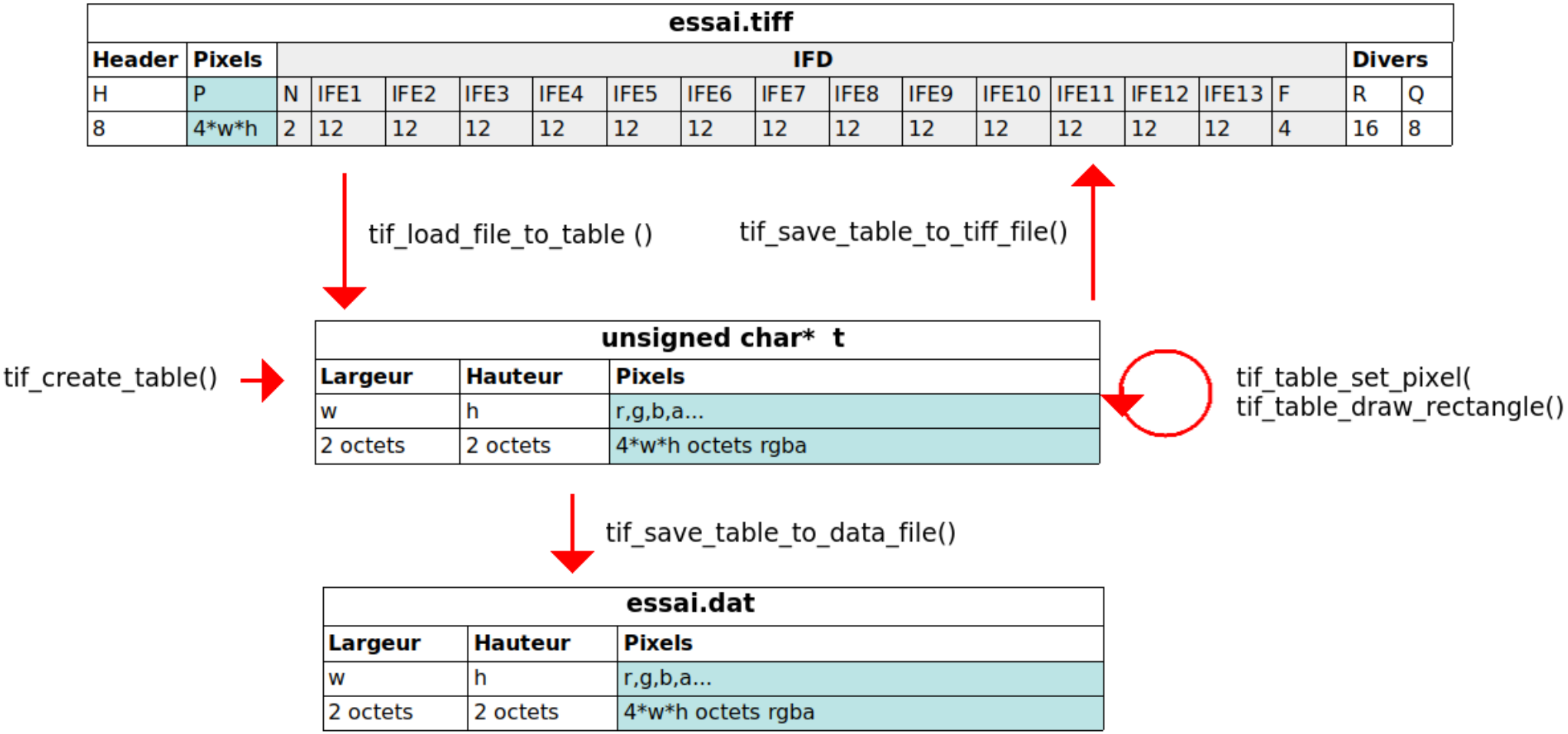
Octets image, dans un fichier .tiff sur le disque dur																		
Header	**Pixels**	**IFD**															**Divers**	
H	P	N	IFE1	IFE2	IFE3	IFE4	IFE5	IFE6	IFE7	IFE8	IFE9	IFE10	IFE11	IFE12	IFE13	F	R	Q
8	4*w*h	2	12	12	12	12	12	12	12	12	12	12	12	12	12	4	16	8

Dans une table t et dans un fichier de données brutes, on ne conserve que les valeurs de la largeur, la hauteur et des composantes de couleur r,g,b et a des pixels de l'image.

Octets image, dans une table t en mémoire		
Largeur	**Hauteur**	**Pixels**
w	h	r,g,b,a...
2 octets	2 octets	4*w*h octets rgba

Octets image, dans un fichier .dat sur le disque dur		
Largeur	**Hauteur**	**Pixels**
w	h	r,g,b,a...
2 octets	2 octets	4*w*h octets rgba

Les fonctions C opèrent de la façon suivante sur les fichiers tiff, sur les tables en mémoire et sur les fichiers de données brutes.



Exemple

- essai.tiff : fichier de départ
  (image rgba, non compressée dont les octets sont arrangés selon le type little endian)

unsigned char *t=NULL;
t=tif_load_file_to_table("essai.tiff",0);↓

- Table t en mémoire

```
00000000 14 00 05 00 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000039 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000078 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF
00000117 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF
00000156 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000195 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000234 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF
00000273 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF
00000312 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000351 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000390 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF
```

tif_save_table_to_tiff_file (t, "test.tiff", 0);↓

- Fichier de destination enregistré sur le disque dur

```
00000000 49 49 2A 00 98 01 00 00 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000039 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000078 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF
00000117 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF
00000156 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000195 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000234 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF
00000273 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF
00000312 FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00
00000351 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00
00000390 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 FF 0D 00 00 01 03 00 01 00 00 00 14 00 00 00 01 01 03 00 01 00 00
00000429 00 05 00 00 00 02 01 03 00 04 00 00 00 4A 02 00 00 03 01 03 00 01 00 00 00 01 00 00 00 06 01 03 00 01 00 00 00 02 00
00000468 00 00 11 01 04 00 01 00 00 00 08 00 00 00 15 01 03 00 01 00 00 00 04 00 00 00 16 01 03 00 01 00 00 00 05 00 00 00 17
00000507 01 04 00 01 00 00 00 90 01 00 00 1A 01 05 00 01 00 00 00 3A 02 00 00 1B 01 05 00 01 00 00 00 42 02 00 00 28 01 03 00
00000546 01 00 00 00 02 00 00 00 52 01 03 00 01 00 00 00 02 00 00 00 00 00 00 00 48 00 00 00 01 00 00 00 48 00 00 00 01 00 00
00000585 00 08 00 08 00 08 00 08 00
```