TP 5 – miniprojet - tubes et fichiers

⟶ *Vademecum* : je compile, je vérifie que les exe n'ont pas changé de nom dans les execl, je donne un usage, je mets des erreurs explicites, je fais un readme pour indiquer quoi lancer si plusieurs fichiers ⟵

Mini-projet :

- ☞ Cet exercice sera à déposer sur ARCHE pour le **jeudi 7 mai 2020 avant 23h55**.
- ☞ Outre le programme lui-même, la qualité de la programmation, des commentaires et l'affichage lors de l'exécution du programme est important
- ☞ Vous travaillez pour vous (et notamment pour le semestre qui suivra)

L'objectif du programme est de créer une copie d'un fichier original au format BMP et de créer en parallèle un processus qui convertira l'image en niveau de gris (et différentes couleurs). La transmission des données se fera par tube anonyme. Il est conseillé de procéder étape par étape :

[dans les premières étapes, il n'est pas encore question de tube.]

- Réalisez un programme qui teste l'existence d'un fichier (donné en argument), l'ouvre puis crée un nouveau fichier et recopie octet par octet le fichier d'entrée dans ce dernier. Vous utiliserez les fonctions `stat`, `read` et `write`.

aide C :

- ☞ Pour l'ouverture d'un fichier en écriture afin de le créer utiliser les constantes `O_WRONLY | O_CREAT | O_TRUNC` dans la fonction `open` (avec S_IRWXU en troisième argument).
- ☞ Pour l'ouverture d'un fichier en lecture... c'est O_RDONLY sans 3ème argument.
- ☞ N'oubliez pas de fermer le fichier.

- Modifiez ce programme pour qu'il vérifie que le fichier d'entrée soit bien une image BMP. Il s'agit de vérifier que les 2 premiers octets sont 'B' et 'M'. (n'oubliez pas de continuer à réécrire directement les octets dans le nouveau fichier)
- À l'aide du tableau ci-dessous qui donne la structure d'un en-tête de fichier BMP, retrouvez les informations suivantes en les affichant : résolution de l'image (largeur et hauteur), le nombre de bits par pixel, et si le fichier est compressé. Vous procéderez en avançant objet

par objet (char, int, short...) avec la fonction `read` (et non plus octet par octet) .

Données	**Type**	**Valeur**	**Description**
entête du fichier			
file_type	C[2]	'BM'	identification du format (d'autres existes pour OS/2 principalement)
file_size	DW		taille du fichier
reserved	DW	0	réservé (inutilisé)
bitmap_offset	DW		offset de l'image (=adresse des données de l'image (sans l'entête))
entête du bitmap			
header_size	DW		taille de l'entête en octets
width	DW		largeur en pixels de l'image
height	DW		hauteur en pixels de l'image
planes	W	1	nombre de plans utilisés
bits_per_pixel	W	1,4,8,16,24,32	nombre de bits utilisés par pixel
compression	DW	0, 1, 2, 3	0 = pas de compression
size_bitmap	DW		taille de l'image en octets
horiz_resolution	DW		résolution horiz. en pixels par mètre
vert_resolution	DW		résolution vert. en pixels par mètre
colors_used	DW		0= palette entière
colors_important	DW		nombre de couleurs importantes
palette (optionnelle)			la liste des correspondances

C[2] = 2 caractères (2 char).
W = 1 mot : 2 octets (1 short) .
DW = 2 mots : 4 octets (1 int).

- Nous allons maintenant réaliser l'image en niveau de gris. Il faut pour cela identifier le début réel de l'image (retrouvez le champ qui le donne dans l'en-tête). Au lieu d'utiliser une compteur incrémenté à chaque lecture pour savoir à tout instant sur quel octet on est, nous allons utiliser la fonction `lseek` (juste après la récupération de l'information de la compression) pour retrouver la position courante. Ensuite il suffit d'avancer jusqu'au début de l'image (ce qui nous évitera de rechercher des information sur la taille de la palette ou d'autres informations pouvant avoir été insérées).

aide C :

☞ Pour obtenir la position courante on exploite la valeur de retour de la fonction `lseek`. Cette fonction sert normalement à se déplacer dans le fichier, à partir d'un point donné (début SEEK_SET, courant SEEK_CUR, fin SEEK_END) et renvoie la valeur de la nouvelle position.

- Votre programme ne traitera que des fichiers BMP 24 bits. Le programme ne devra pas faire le traitement suivant si le pixel n'est pas codé sur 24 bits (afficher un message d'erreur).

- La composition d'une image est une succession de ligne de pixels (la première ligne est en bas de l'image). Un pixel codé sur 24 bits est donc composé de 3 octets codant dans l'ordre : Bleu, Vert, Rouge. Petite particularité : chaque ligne doit être un multiple de 4 octets. Si le nombre de pixels en largeur multiplié par 3 octets n'est pas un multiple de 4, la ligne est complétée par des zéros (vous pouvez utiliser l'opérateur *modulo* % pour connaitre le nombre d'octets restant à mettre à 0). Pour mettre en niveau de gris, bien que le plus simple soit de prendre les valeurs des couleurs, de faire la moyenne et de donner cette valeur aux 3 octets finaux. Vous utiliserez la formule pour un meilleur rendu (pour chaque octet, utilisez un `unsigned char` pour que la valeur soit comprise ente 0 et 255) :
```
pixelGris = (char)(pixelR * .299 + pixelV * .587 + pixelB * .114);
newpixelB=pixelGris; newpixelV=pixelGris; newpixelR=pixelGris;
```
- Testez le comportement de votre programme avec différents fichiers BMP ou autres.
- Réalisez l'objectif final :
  - Le père ouvre un fichier BMP, crée un tube et un fils. Lors de la lecture du fichier, le père affichera toujours les informations précédemment demandées et écrira dans un fichier (pour réaliser une copie du fichier) et aussi dans le tube pour envoyer les octets au fils. Le nom du fichier à générer sera donné en argument du père. Ex : `prog fichier.bmp fichier_final.bmp`. Le fichier est envoyé dans le tube **octet par octet** au fils.
  - Le fils prendra en charge le traitement en niveau de gris (et créera donc un nouveau fichier). Le fils **n'ouvre pas le fichier BMP**, il reçoit les octets par le tube (il s'agit quasiment d'un copié-collé du code précédent...).
  - Pour faciliter le traitement le père pourra envoyer à son fils (avec *execl*) la taille de l'entête du fichier ainsi que la largeur de l'image, pour éviter au fils de tout redécoder pour savoir où se situent les premiers pixels ou traiter les octets de fin de ligne. Il enverra aussi le nom du fichier final. Le fils le renommera en ajoutant "_gris" avant l'extension. Ex : fichier_final.bmp deviendra fichier_final_gris.bmp.
- Objectif final 2 :
  - Si (et seulement si) le premier objectif est atteint. Complétez le programme pour que 3 autres fils s'occupent de générer un fichier respectivement en niveau de rouge, en niveau de vert et en niveau de bleu (=on ne garde que la composante de la couleur, les autres sont mises à 0).
  - Il faut donc générer en tout 4 fils et 4 tubes.