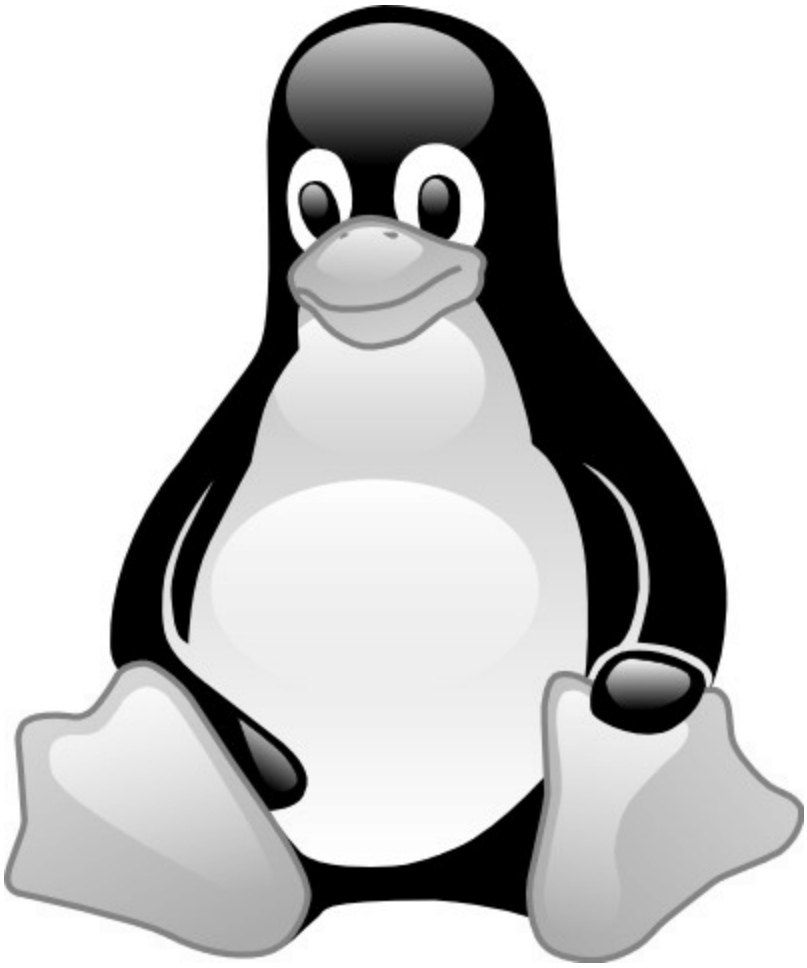
**Maxime Soulié**

# 1: niveaux de gris

transformer une image en niveaux de gris, il faut que toutes les composantes d'un pixel aient les memes valeurs. pour cela, j'utilise l'equation `c = 0.299\*RED + 0.587\*GREEN + 0.114\*BLUE` et assigne a chaque composante de pixel la valeur c

```
def greyscale(path):
    pixels = img.read_img(path)
    for i in pixels:
        for j in i:
            c=0.299*j[0]+0.587*j[1]+0.114*j[2]
            j[0]=j[1]=j[2]=c
    img.write_img(path[:len(path)-4]+"_grey.png",pixels)
    img.display_img(pixels)
```

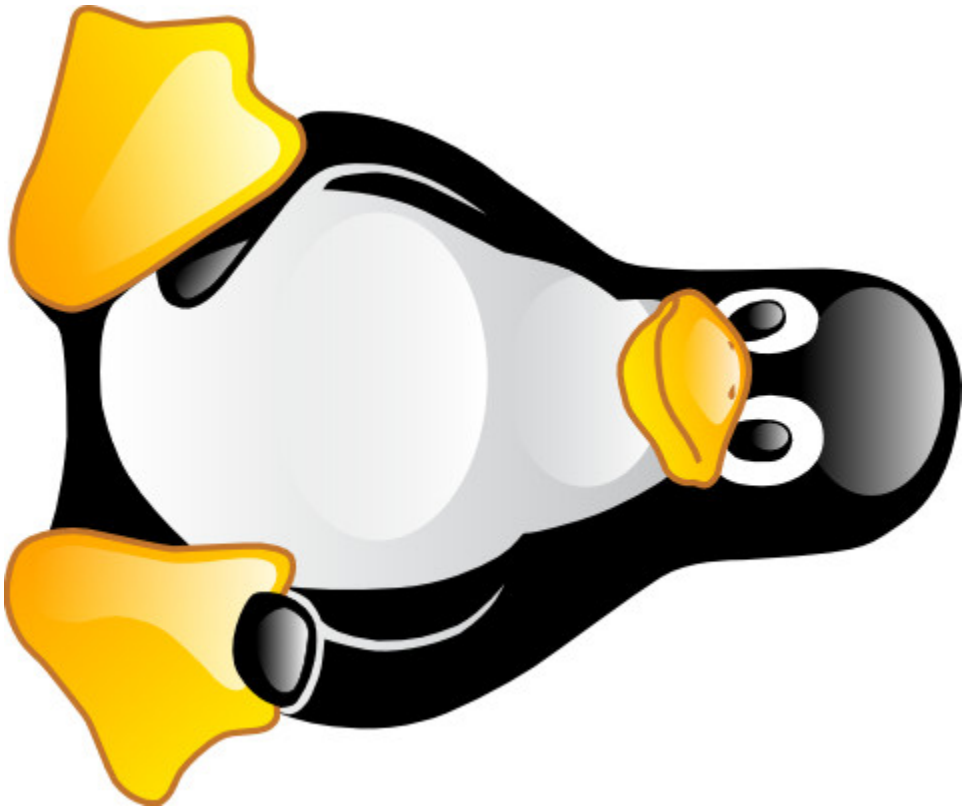


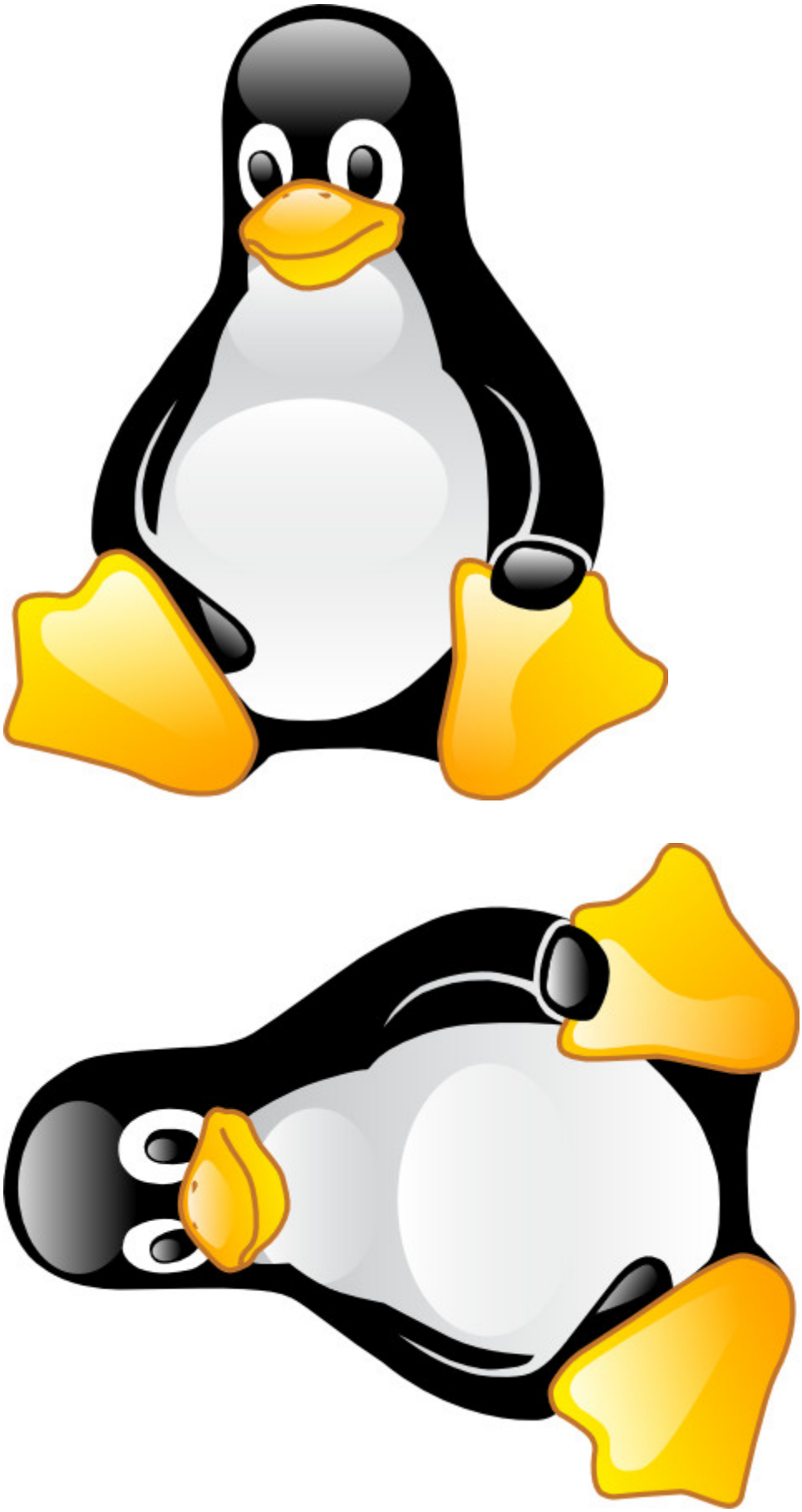
TUX.png en niveaux de gris

# 2: rotations

afin de faire les rotations, j'ai decidé d'utiliser la transposée de la matrice des pixels. cepandant, utiliser simplement la transposée nous donne une image avec une retoation et un symetrie, il faut donc "lire" cette matrice en sens inverse sur l'un des axes, selon la direction de l'inversion.
pour faire la transposée, il est possible de proceder ainsi :

```
def transpose(array):
    if array.ndim != 2:
        return("Error: array must be 2D")
    transposed = np.zeros(array.shape)
    for i in range(len(array)):
        for j in range(len(array[0])) :
            transposed[j][i] = array[i][j]
    return transposed
```





toutes les rotations possibles de TUX.png

Cepandant, pour des raisons d'optimisation, j'ai utilisé la transposée fournie par le module numpy (cette bibliotheque étant écrite en C, elle est plus rapide que du python).

```
def rotate(path, dir):
    pixels = img.read_img(path)
    print(pixels)
    if dir=="-":
        pixels=img.read_img(path).transpose(1,0,2)[::-1]
        img.write_img(path[:len(path)-4]+"_rotated-90.png",pixels)
    elif dir=="+":
        pixels=img.read_img(path).transpose(1,0,2)[:,::-1]
        img.write_img(path[:len(path)-4]+"_rotated+90.png",pixels)
    elif dir=="0":
        pixels=img.read_img(path)
        img.write_img(path[:len(path)-4]+"_rotated-0.png",pixels)
    elif dir=="180":
        pixels=img.read_img(path)[::-1,::-1]
        img.write_img(path[:len(path)-4]+"_rotated-180.png",pixels)
    else:
        print("Error: dir must be '+', '-', '0' or '180'")
        return
    img.display_img(pixels)
```

# 3: stéganographie

Pour la stéganographie, j'ai caché chaque lettre sur la ligne correspondante a sa position sur la chaine de caractere, et sur la colonne correspondant a son numero ( `ord(c)-97` 97 etant l'indice de ). j'ai modifié une comnposante aleatoire de ce pixel de -1 (ou +1 si la composante est a 0). Pour décoder le message avec l'image de reference, je fais la soustraction des 2 matrices. si une composante n'est pas a 0, je recupere le numero de la colonne. La lettre correspondante est `chr(col+97)` .

la fonction stegano verifie aussi si la chaine ne contient que des caracteres de l'alphabet (espaces non compris), et si la longueur de la chaine ne depasse pas 50.

```
def stegano(string, iamgePath):
    if len(string)>50:
        print("string must be not more than 50 characters")
        return
    if not string.isalpha():
        print("string must be only letters (no space, no punctuation, no numbers nor spec
        return


    string=string.lower()
    chars=[]
    set=np.ndarray([0])
    for c in string:
        chars.append(ord(c))
    pixels = img.read_img(iamgePath)
    for i in range(len(chars)):
        rand=random.randint(0,2)
        if pixels[i][chars[i]-97][rand] == 0:
            pixels[i][chars[i]-97][rand] +=1
        else:
            pixels[i][chars[i]-97][rand] -=1
    img.write_img(iamgePath[:len(iamgePath)-4]+"_stegano.png",pixels)

def destegano(iamgePath, refPath):
    chars=[]
    string=""
    pixels = img.read_img(iamgePath)
    pixelsRef= img.read_img(refPath)
    diff=pixels-pixelsRef
    for i in range(len(diff)):
        for j in range(len(diff[0])):
            if diff[i][j][2] != 0 or diff[i][j][1] != 0 or diff[i][j][0] != 0:
                chars.append(chr(j+97))
    for i in range(len(chars)):
        string+=chars[i]
    return string
```

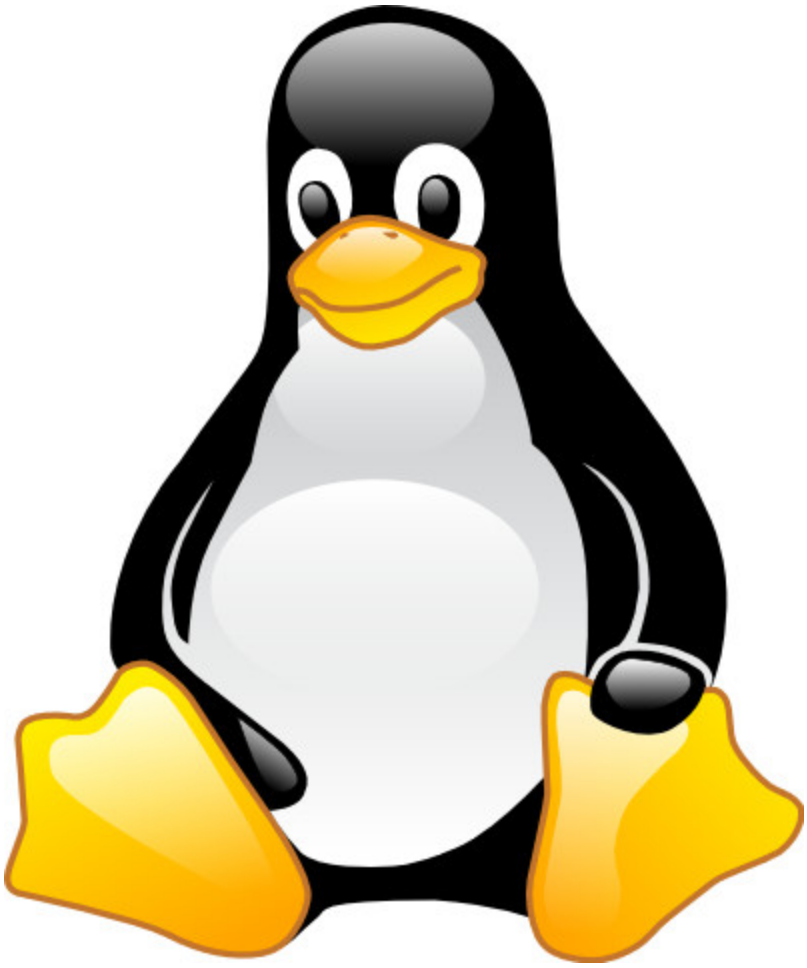


image de reference TUX.png

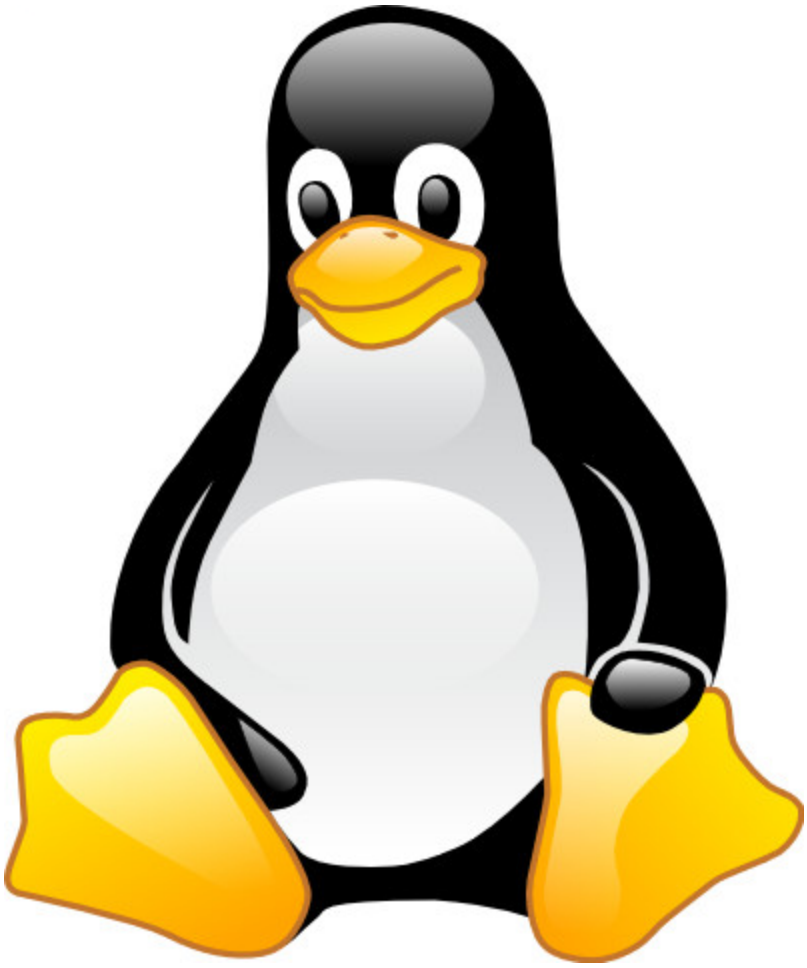


image steganographiée de TUX.png (aucune difference visible a l'oeil nu)