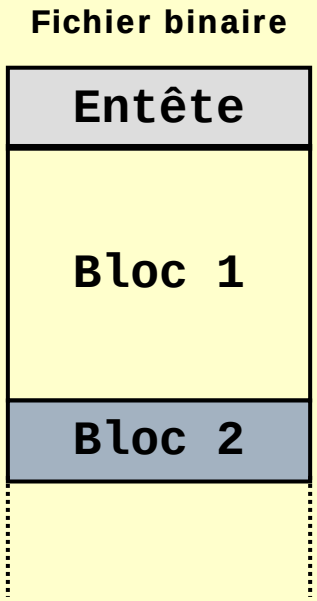
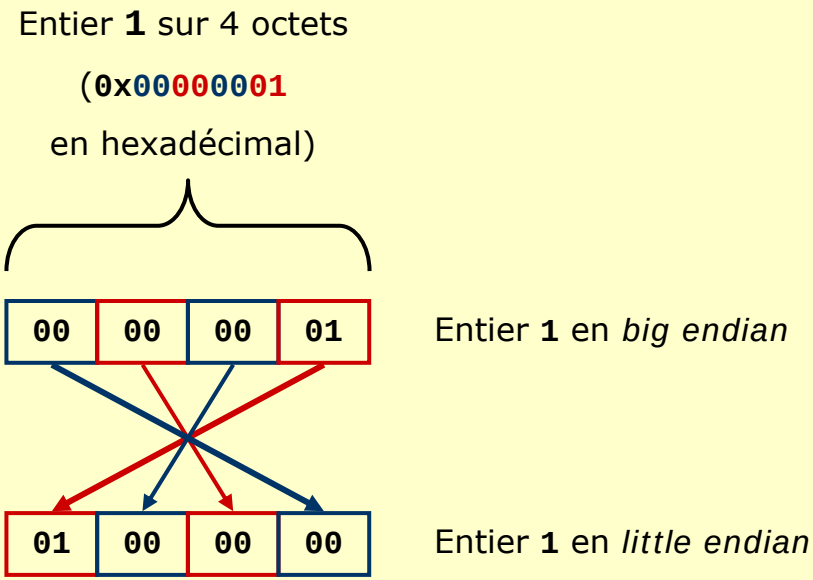
Les fichiers binaires

- Qu'est-ce qu'un fichier binaire (cas général)?
  - pas un fichier texte ☺
    → Pas manipulable avec un éditeur de texte (vi, emacs, …)
  - entête (éventuel) et suite de blocs de données binaires et/ou texte
- Pourquoi utiliser des fichiers binaires?
  - pas pratique à lire… ☹ Sauf si on utilise un format *auto-documenté* (netCDF, …)
  - cela prend moins de place de stocker un nombre en binaire qu'avec des caractères…
    - Entier sur 4 octets : +/- 2^{31} → **-2147483648** **2147483647**
    - Equivalent avec du texte → 11 caractères! 10 caractères
  - la lecture/écriture (par des programmes…) de fichiers binaires est beaucoup plus rapide que celle de fichiers texte



Manipulation de fichiers binaires

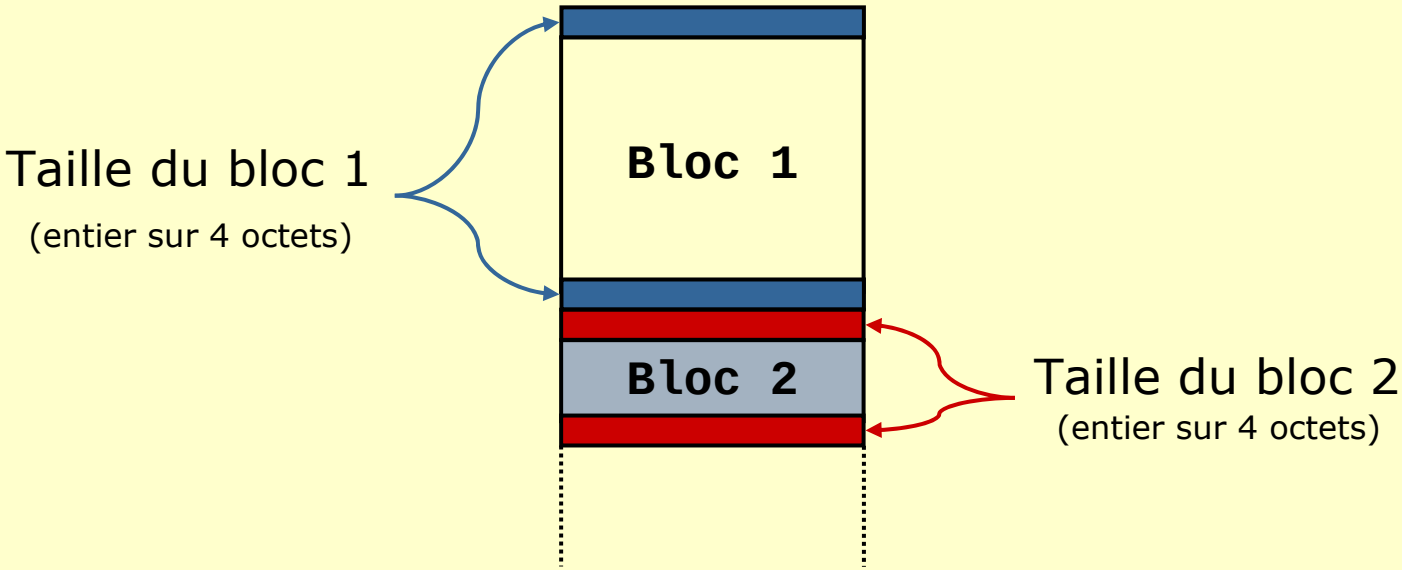
- Comment utiliser un fichier binaire?
  - il faut que le format du fichier soit documenté en détail…
  - …ou disposer du source d'un programme permettant de lire/écrire ce format
  - il faut aussi connaître le type de processeur sur lequel le fichier a été généré pour connaître l'ordre de stockage des octets dans le fichier
    - Intel/AMD, Alpha (DEC) → binaire *little endian*
    - Tous les autres processeurs → binaire *big endian*
    - On passe du *big endian* au *little endian* en retournant l'ordre des octets qui constituent chaque nombre



Fichiers fortran *unformatted* à accès séquentiel…

Fichier binaire fortran *unformatted* à accès séquentiel

- Chaque opération **`WRITE(unit)`** ajoute un bloc de données au fichier (bloc de $\mathbf{nb_i}$ octets)
- Chacun des blocs de données est précédé ET suivi par l'entier $\mathbf{nb_i}$ codé sur 4 octets



Exemple de programme fortran (py_jyp_ex_53.f90)

Ecriture d'un tableau dans un fichier binaire, de 3 façons différentes

```
PROGRAM py_jyp_ex_53

  INTEGER, PARAMETER :: nblignes = 2, nbcolonnes = 3
  INTEGER i, j
  REAL*4, DIMENSION(nbcolonnes, nblignes) :: a

  a(:, :) = 0.
  a(1, :) = 1.
  a(3, :) = 3.
  DO j = 1, nblignes
    WRITE(*, *) a(:, j)
  ENDDO

  OPEN(10, file='a_brut.dat', form='unformatted')
  WRITE(10) a
  CLOSE(10)

  OPEN(10, file='a_entete.dat', form='unformatted')
  WRITE(10) 'a_entete.dat', nbcolonnes, nblignes, a
  CLOSE(10)

  OPEN(10, file='a_entete_multwrite.dat', form='unformatted')
  WRITE(10) 'a_entete.dat'
  WRITE(10) nbcolonnes
  WRITE(10) nblignes
  WRITE(10) a
  CLOSE(10)

END PROGRAM py_jyp_ex_53
```

```
1.000000      0.0000000E+00   3.000000
1.000000      0.0000000E+00   3.000000
```

Initialisation et affichage du tableau à écrire dans les fichiers

Ecriture *simple* du tableau **a**

Ecriture en une seule fois de **a**, précédé par un entête

Ecriture du fichier binaire avec entête, en plusieurs fois

Analyse des 3 fichiers binaires (1/2)

- Commande **od** (*object dump*)
  - od –I fichier → affichage des entiers sur 4 octets
  - od –f fichier → affichage des réels sur 4 octets
  - od –c fichier → affichage des caractères
- Commande **strings**
  - affichage uniquement des caractères visibles

24 octets (*tableau* ***a***) + 2 entiers sur 4 octets (binaire fortran…) = 32 octets

24 octets + 12 octets (*'**a_entete.dat**'*) + 2 entiers sur 4 octets (***nbcolonnes, nblignes***) + 2 entiers sur 4 octets = 32 octets

```
> ls -l a_*.dat
       32 a_brut.dat
       52 a_entete.dat
       76 a_entete_multwrite.dat
```

2 * 3 points * 4 octets = 24 octets

```
> od -f a_brut.dat
0000000   3.363116e-44   1.000000e+00   0.000000e+00   3.000000e+00
0000020   1.000000e+00   0.000000e+00   3.000000e+00   3.363116e-44

> od -l a_brut.dat
0000000           24  1065353216            0  1077936128
0000020  1065353216            0  1077936128           24
```

Analyse des 3 fichiers binaires (2/2)

```
> od -f a_entete.dat
0000000   6.165713e-44   1.774684e+28   7.213306e+22   7.142936e+31
0000020   4.203895e-45   2.802597e-45   1.000000e+00   0.000000e+00
0000040   3.000000e+00   1.000000e+00   0.000000e+00   3.000000e+00
0000060   6.165713e-44

> od -l a_entete.dat
0000000          44  1852137313  1702126964  1952539694
0000020           3           2  1065353216           0
0000040  1077936128  1065353216           0  1077936128
0000060          44

> od -c a_entete.dat
0000000   ,  \0  \0  \0   a   _   e   n   t   e   t   e   .   d   a   t
0000020 003  \0  \0  \0 002  \0  \0  \0  \0  \0 200   ?  \0  \0  \0  \0
0000040  \0  \0   @   @  \0  \0 200   ?  \0  \0  \0  \0  \0  \0   @   @
0000060   ,  \0  \0  \0

> strings a_entete.dat
a_entete.dat
```

```
> od -f a_entete_multwrite.dat
0000000   1.681558e-44   1.774684e+28   7.213306e+22   7.142936e+31
0000020   1.681558e-44   5.605194e-45   4.203895e-45   5.605194e-45
0000040   5.605194e-45   2.802597e-45   5.605194e-45   3.363116e-44
0000060   1.000000e+00   0.000000e+00   3.000000e+00   1.000000e+00
0000100   0.000000e+00   3.000000e+00   3.363116e-44

> od -l a_entete_multwrite.dat
0000000          12  1852137313  1702126964  1952539694
0000020          12           4           3           4
0000040           4           2           4          24
0000060  1065353216           0  1077936128  1065353216
0000100           0  1077936128          24
```