5- LANGAGE C

POINTEUR ET TABLEAU DYNAMIQUE

Loïc Cuvillon

l.cuvillon@unistra.fr

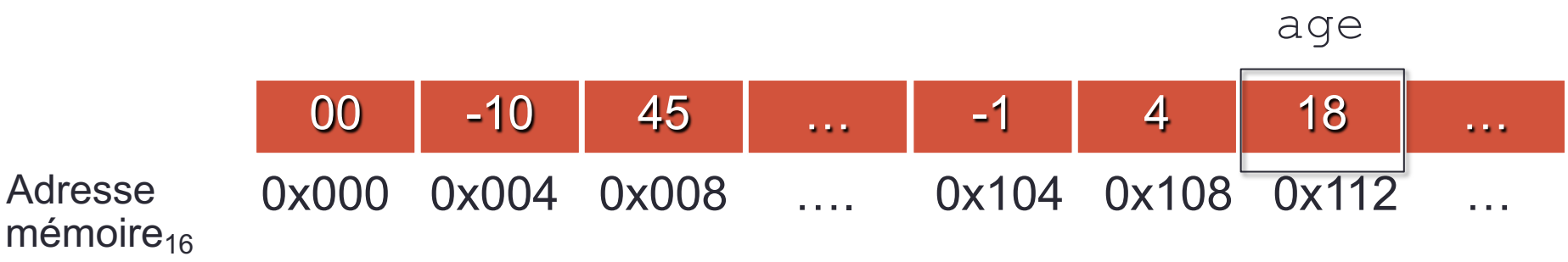
Sommaire chapitre 5

Le pointeur

Rappel : la variable

- un nom `age`
- une valeur `18`
- une adresse mémoire où elle est stockée `0x112`
  (que l'on obtient avec l'opérateur adresse: `&`)

```
printf("%d", age);    →   18
printf("%p", &age);   →   0x112
```



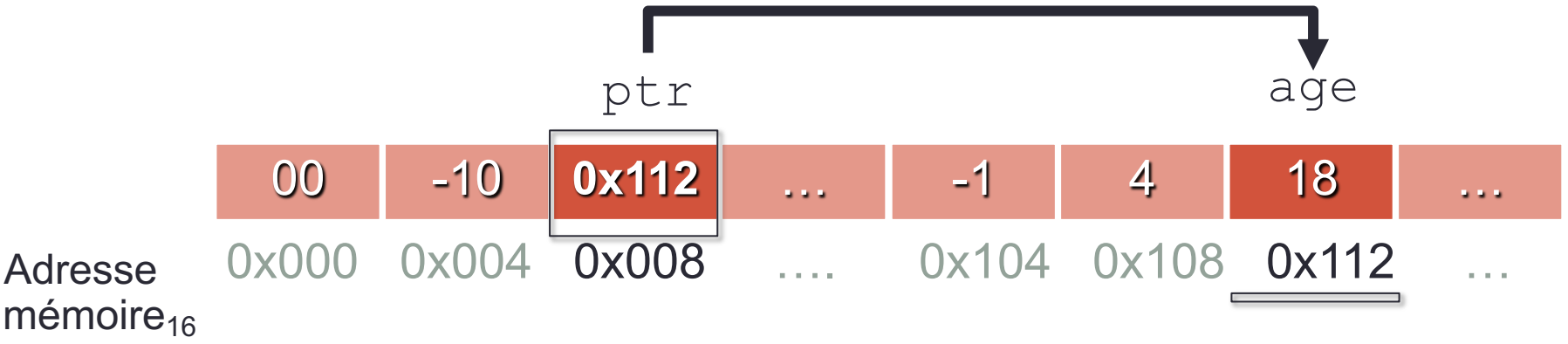
Le pointeur

Un pointeur est une variable qui contient une adresse mémoire.

`ptr = &age;` affecte l'adresse de `age` à la variable `ptr`.

On dit que `ptr` :

- `ptr` contient l'adresse de la variable `age`.
- `ptr` est un pointeur/pointe sur `age`.

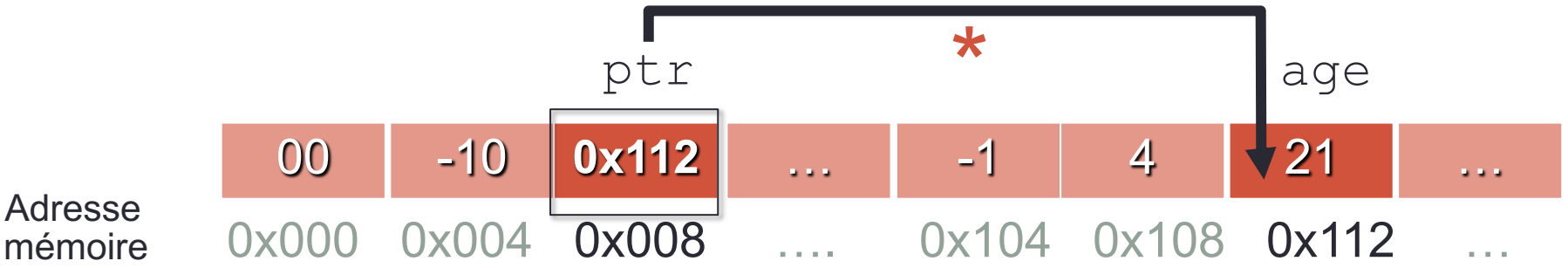


Opérateur d'indirection

A partir de l'adresse (stockée dans un pointeur), on accède à la variable pointée via * : opérateur d'indirection ou de déréférencement

On peut dire que * est l'opérateur "variable pointée par".

```
*ptr = 21 ;                    /* mettre 21 dans la variable
                                  pointée par ptr, soit age */

printf("%d", *ptr);            /* affiche 21 : valeur de
                                  la variable pointée */
```



Déclaration d'un pointeur

Déclaration en fonction du `type` de la variable pointée :

```
type *nom_du_pointeur;
```

```
int *ptr;
```

- déclaration d'un pointeur `ptr` sur un entier **`int`**
- `ptr` et `&age` ont tous deux pour type: **`int`**`*`

Pointeur sur un réel: **`double *f ;`**

sur un caractère: **`char *c;`**

sur un type non connu: **`void *generic;`**.

(une fois le `type` connu, utilisable via l'opérateur de conversion de type `(type*)`)

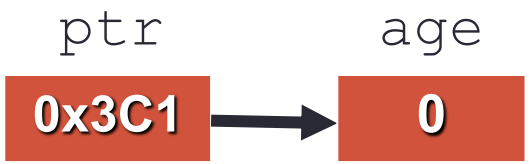
Initialisation d'un pointeur

- Initialisation : avec l'adresse d'une variable

```
int age=0;
int *ptr;
ptr = &age;
```

↔

```
int age=0;
int *ptr = &age;
```



ou

- tant qu'il ne pointe sur rien,

toujours initialiser avec le pointeur nul (adresse nulle) : **NULL**.

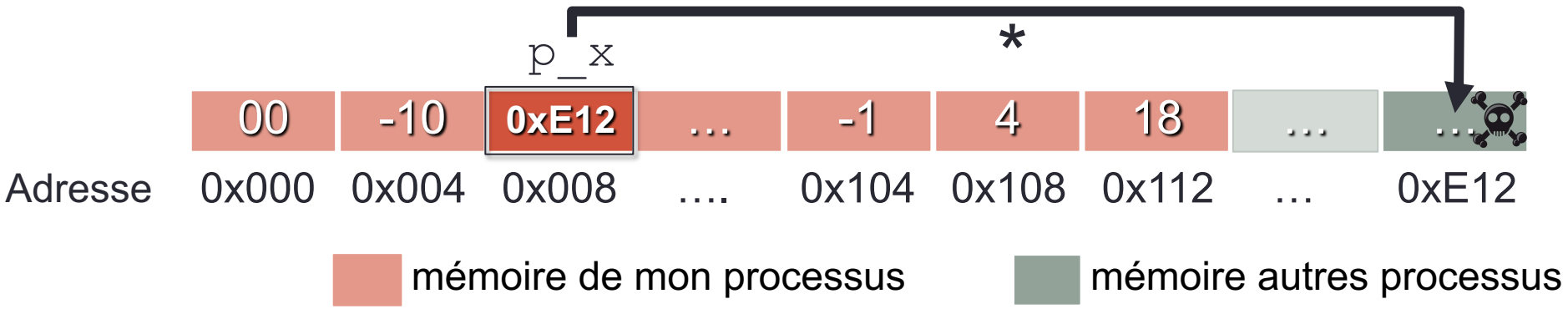
```
int *ptr = NULL;
```

Initialisation d'un pointeur

- En l'absence d'initialisation correcte:

```
int *p_x ;      /*p_x contient une adresse aléatoire!*/
*p_x = 3 ;
```

1/ Si l'adresse n'appartient pas à la mémoire réservée pour notre programme:



Tentative d'écriture de la valeur 3 hors de la mémoire autorisée (`*p_x=3`) → le système d'exploitation l'interdit et tue immédiatement mon programme en exécution (`Segmentation Fault`) pour préserver l'intégrité du système.

Initialisation d'un pointeur

2/ Si l'adresse aléatoire appartient au programme (le pire cas) :

la mémoire du programme est corrompue, insidieusement la valeur d'une des variables du programme est définitivement perdue, écrasée par 3 (`*p_x=3`).

Sommaire chapitre 5

Pointeurs

- Opération sur la variable pointée (via l'opérateur d'indirection) :
  toute opération permise avec la variable pointée

```
int *pi = &x;              /* *pi pointe sur x */

*pi = *pi + 2;             /* <->  x = x + 2   */

y = *pi - 1;               /* <->  y = x - 1   */

(*pi)++;                   /* <->  x++         */
```

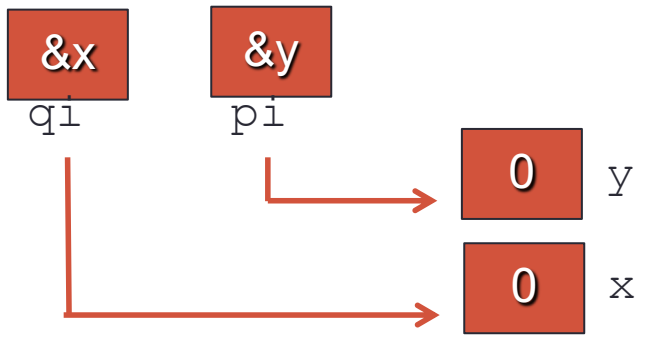
Pointeurs

- Opération sur le pointeur lui-même (sans indirection) :

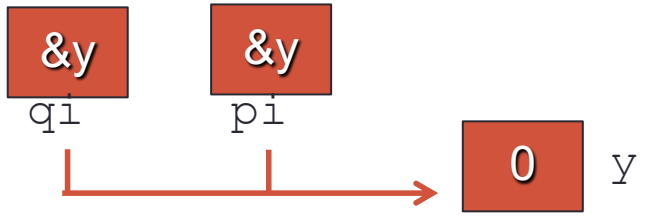
car les pointeurs sont des variables

① L'affectation :

```
int *qi=&x, *pi=&y;
```




```
qi = pi;        /* qi contient même adresse que pi   */
                /* qi, pi pointent sur même variable */
```



Pointeurs

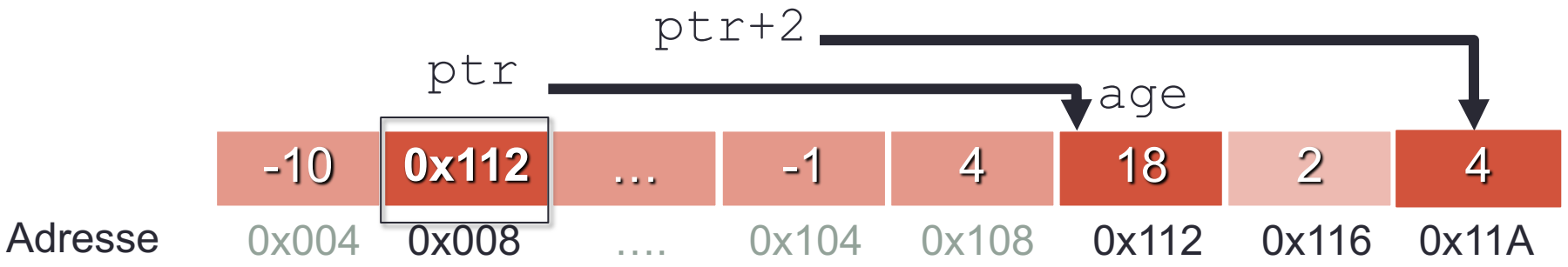
② Arithmétique entière sur un pointeur:

Soit un pointeur `p` de type `type *` et un entier `i`, alors

- `p+i` est encore un pointeur de type `type *`
- `p+i` vaut `p` incrémenté de `i * sizeof(type)`.

→ `p+i` est l'adresse de la `i`ème variable en mémoire de même `type` derrière celle pointée par `p`.

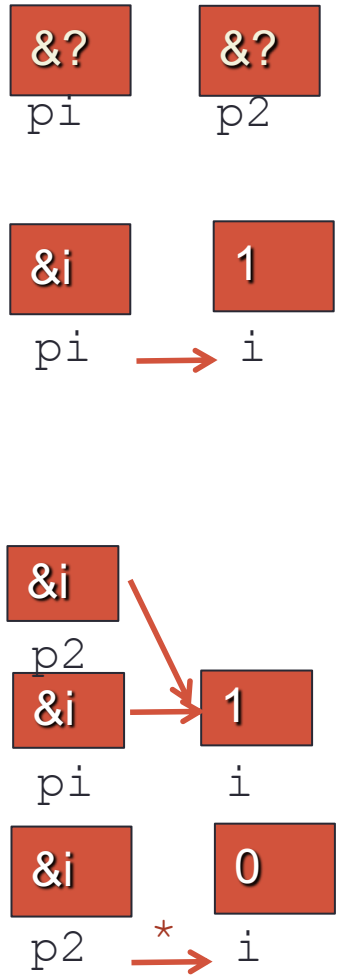
`ptr+2` vaut avec `sizeof(int)`=4, `0x112 + 2*4 = 0x11A`



Pointeurs

- Exemple récapitulatif :

```
int i = 1, j = 2 ;

int *pi, *p2;    /* pi, p2 pointent sur un int */

pi = &i;         /* pi pointe maintenant sur i */

j = *pi;         /* j vaut désormais 1 */

p2=pi;          /*p2, pi pointent sur même*/

*p2 = 0;         /* i vaut désormais 0 */
```



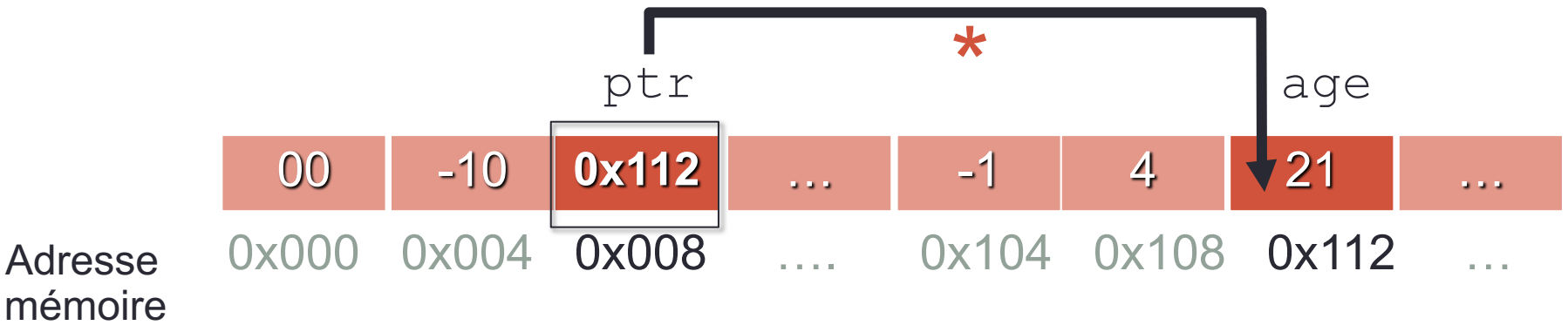
En résumé

- Le pointeur :

1. stocke une adresse mémoire

1. permet l'accès indirect (opérateur `*`) à la variable située à cette adresse

2. doit être initialisé avec une adresse correcte avant utilisation !




MAN, I SUCK AT THIS GAME.
CAN YOU GIVE ME
A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.

xkcd.com

Mais à quoi ça sert un pointeur ?

1. À créer des tableaux dynamiques, dont la taille n'est pas connu avant l'exécution du programme (suite du chapitre)

2. À permettre à une fonction de modifier une variable appartenant à la fonction appelante (chapitre 6)