

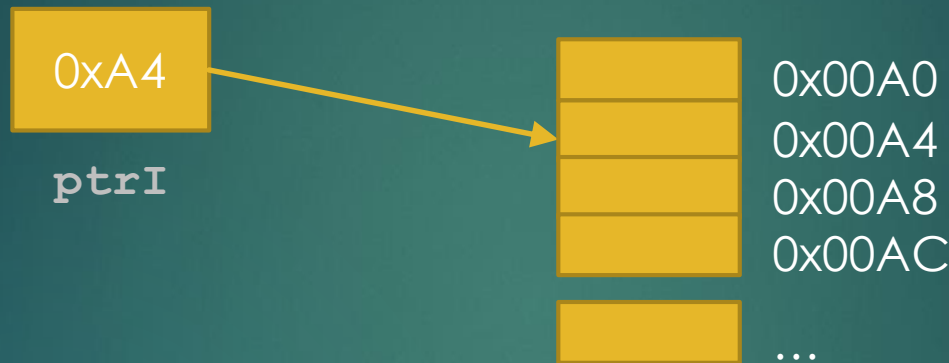


Chap. 4: Les pointeurs

Les pointeurs

2

- Les pointeurs contiennent des adresses mémoires



- Les pointeurs sont typés

```
int *ptrI;    // pointeur vers un entier
```

```
double *ptrI; // pointeur vers un double
```

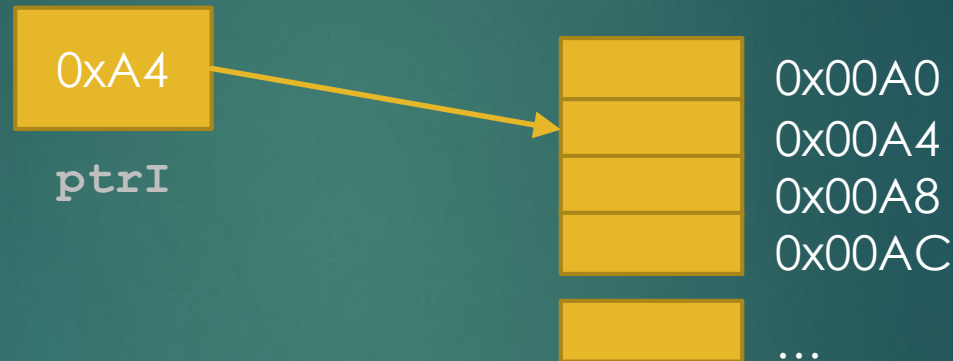
```
Noeud *ptrN;  // pointeur vers un Noeud
```

- Un peu comme une référence Java mais...

Ca sert à quoi?

3

- ▶ Accéder une adresse précise (driver, mémoire vidéo...)



- ▶ Allouer dynamiquement de la mémoire
- ▶ Créer des structures de données chaînées
- ▶ Passer des paramètres par référence

Initialisation

- ▶ Pas initialisé par défaut. Danger!
- ▶ Prendre l'adresse d'une variable

```
int a = 38;
```

```
int *ptrI = &a; // reçoit l'adresse de a
```



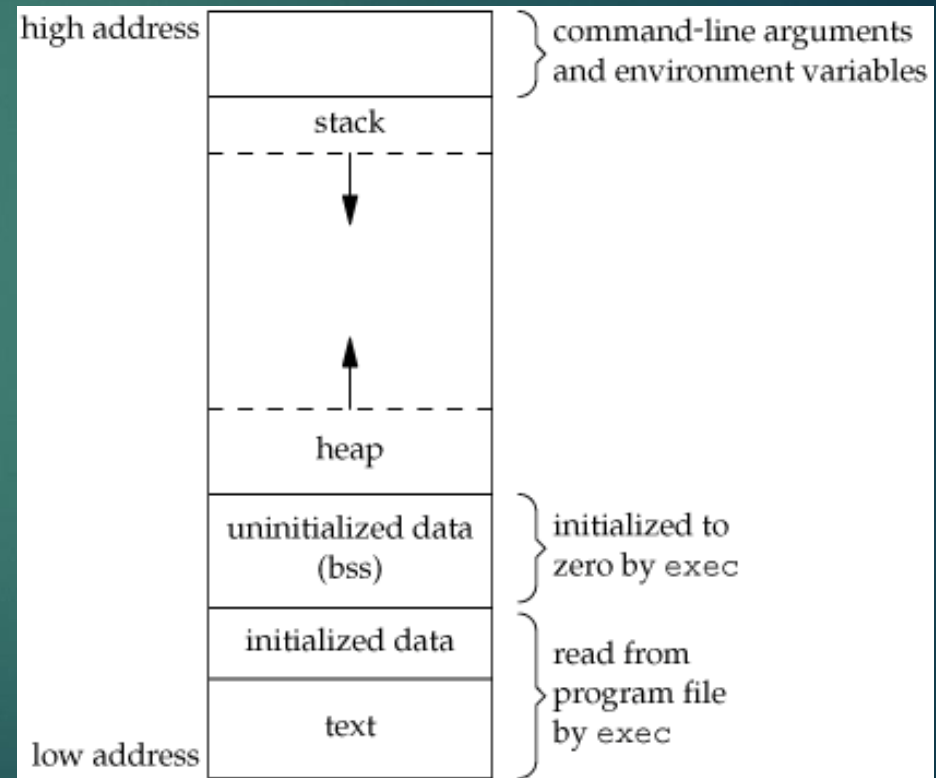
- ▶ Vous avez déjà utilisé le '&'. Où ça?

Initialisation

5

```
int truc() {  
    int a = 38;  
    int *ptrI = &a;  
}
```

► a est sur le **stack**



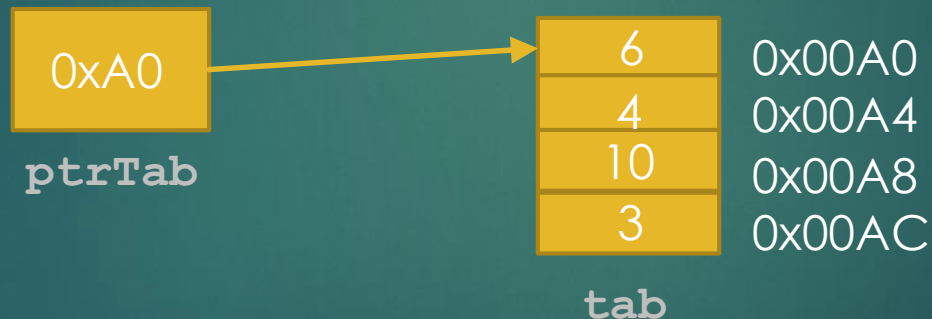
Initialisation

6

- Prendre l'adresse d'un tableau

```
int tab[4] = {6, 4, 10, 3};
```

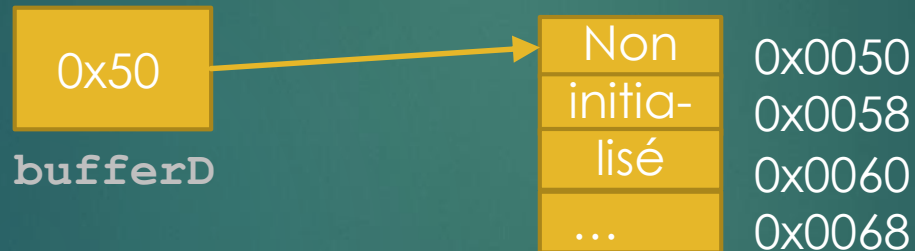
```
int *ptrTab = tab; // adresse de l'élément 0
```



Mémoire dynamique

- Allouer une zone mémoire de taille quelconque

```
double *bufferD =  
    (double*) malloc (4 * sizeof (double)) ;
```



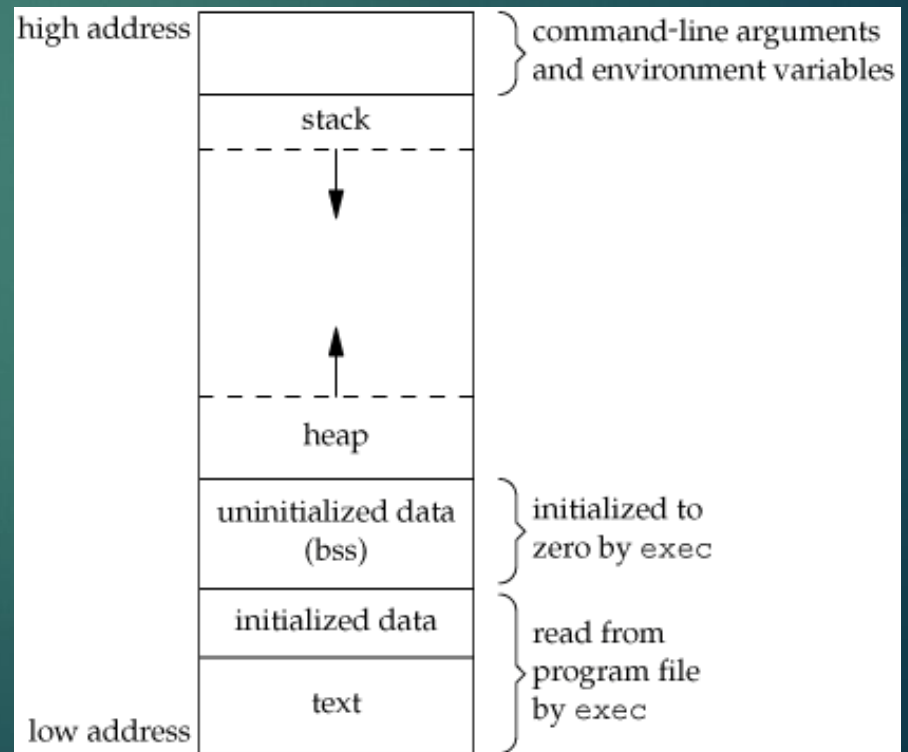
- Remarquez les +8 dans les adresses. Pourquoi?

Mémoire dynamique

- Allouer une zone mémoire de taille quelconque

```
double *bufferD =  
    (double*) malloc (4 * sizeof (double)) ;
```

- malloc travaille sur le heap



Mémoire dynamique

9

- ▶ Si à court de mémoire

```
double *bufferD =  
    (double*)malloc(4*sizeof(double));  
  
if (bufferD == NULL) ...
```



Mémoire dynamique

10

- ▶ Pas de garbage collector
- ▶ Libérez la mémoire!

```
double *bufferD =  
    (double*) malloc (4*sizeof(double)) ;  
  
...  
  
free (bufferD) ;
```

Déréférencement

11

- Accéder au contenu de l'élément pointé

```
int a = 38;
```

```
int *ptrI = &a; // reçoit l'adresse de a
```

```
int b = *ptrI; // déréférencement: prend le  
              // contenu de l'élément pointé
```



Déréférencement

12

- Modifier le contenu de l'élément pointé

```
int a = 38;
```

```
int *ptrI = &a; // reçoit l'adresse de a
```

```
*ptrI = 27; // modifie le contenu de a!
```

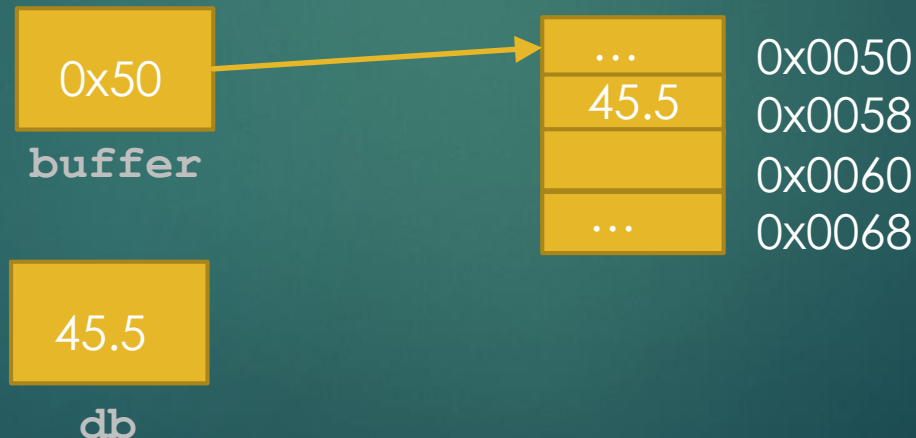


Déréférencement

```
double *buffer =  
    (double*)malloc(4*sizeof(double));
```

...

```
double db = buffer[1]; // déréférencement: accès  
                       // au contenu de l'élément 1
```



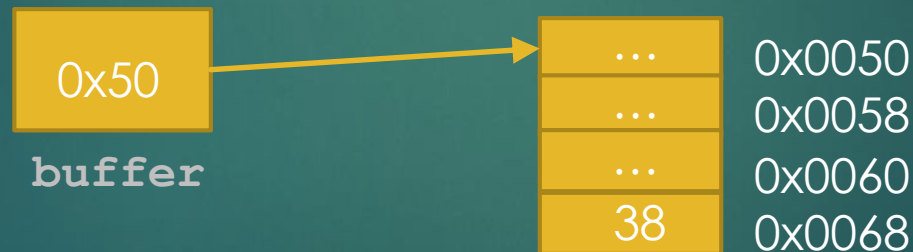
Déréférencement

14

```
double *buffer =  
    (double*)malloc(4*sizeof(double));
```

...

```
buffer[3] = 38; // modifie l'élément 3
```



Déréférencement

15

Programmer's view of matrix

	Col 1	Col 2	Col 3	Col 4
Row 1	1, 1	1, 2	1, 3	1, 4
Row 2	2, 1	2, 2	2, 3	2, 4
Row 3	3, 1	3, 2	3, 3	3, 4

Storage for Matrix – Right-most Index Varies Fastest Scheme

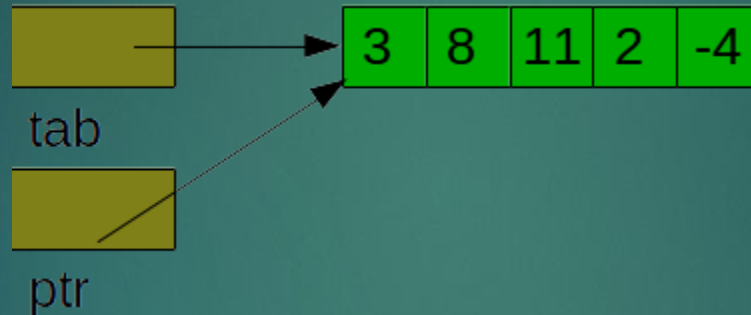
1	2	3	4	5	6	7	8	9	10	11	12
1, 1	1, 2	1, 3	1, 4	2, 1	2, 2	2, 3	2, 4	3, 1	3, 2	3, 3	3, 4

Arithmétique des pointeurs

16

```
int tab[5] = {3, 8, 11, 2, -4};
```

```
int *ptr = tab;
```



```
ptr++;
```

