

# Programmation Avancée S6 IMA3

C avancé : Pointeurs de fonctions

Laure Gonnord

<http://laure.gonnord.org/pro/teaching/>  
Laure.Gonnord@polytech-lille.fr

Université Lille 1 - Polytech Lille

Mars 2011



## Rappels sur les prototypes de fonctions

**Prototype = déclaration** précisant :

- le **nom** de la fonction,
- son **type de retour**, ou `void`,
- le nombre et le **type de ses arguments**.

Exemples :

```
int additionne(int x, int y);
void affiche(char*);
```

Différences avec une définition de fonction :

- le corps est omis (remplacé par ;),
- le nom des arguments est facultatif,
- une fonction peut être déclarée plusieurs fois.

## Rappels sur les prototypes de fonctions

**Utilité du prototype :**

information nécessaire pour compiler un **appel** de fonction.

Si un prototype est donné, la fonction peut être définie

- après l'appel, ou
- dans une autre unité de compilation.

**Exemple :**

Fichier a.c

```
int fun1(int x);

int fun2(void)
{ return fun1(1); }

int fun1(int x)
{ return x+1; }
```

Fichier b.c

```
int fun2(void);

int main()
{ return fun2(); }
```

## Pointeurs de fonctions

Chaque fonction a une adresse en mémoire.

**Pointeur de fonction = variable**

- contenant l'**adresse** d'une fonction,
- dont le type indique le **prototype de la fonction**.

**Déclaration** d'un pointeur de fonction ptr :

```
type-ret (*ptr)(type1, ..., typeN);
```

ptr peut contenir l'adresse de toute fonction

- prenant N arguments, de types type1 à typeN, et
- retournant une valeur de type type-ret.

## Affectation de pointeur de fonction

**Affectation** : `ptr = f;` où

- `f` est le nom d'une fonction de prototype compatible avec `ptr`,
- `f` est un pointeur de fonction de même type que `ptr`.

Exemple :

```
int truc(int x) { ... }

int bidule(int y);

void main()
{
    int (*ptr)(int);
    ptr = truc;
    ptr = bidule;
}
```

## Appel de fonction par pointeur

**Appel par pointeur** : identique à un appel de fonction classique

- `ptr(arg1, ..., argN);` ou
- `x = ptr(arg1, ..., argN);`

Exemple d'appel :

```
int bidule(int y);

int main()
{
    int (*ptr)(int);
    ptr = bidule;
    return ptr(2);
}
```

## Application : fonction d'ordre supérieur

Fonction **paramétrée par une fonction**.

► on passe un pointeur de fonction en argument.

```
void affiche_tableau(char* tab[], void (*f)(char*), int n)
{
    int i;
    for (i=0; i<n; i++) f(tab[i]);
}

void affiche(char* s) { printf("%s\n", s); }

void main() {
    char* t[] = { "toto", "titi" };
    affiche_tableau(t, affiche, 2);
}
```

## Applications

Faire les deux exos suivants :

- Écrire une fonction `fois2` qui réalise la fonction  $x \mapsto 2x$ . Écrire une fonction `appliquer_traitement_tab` qui applique une fonction `f` sur un tableau `T` (d'entiers) de taille `n`. Recoller les bouts.
- Réaliser une fonction de tri générique de tableau qui prend en argument une fonction de comparaison de deux entiers.