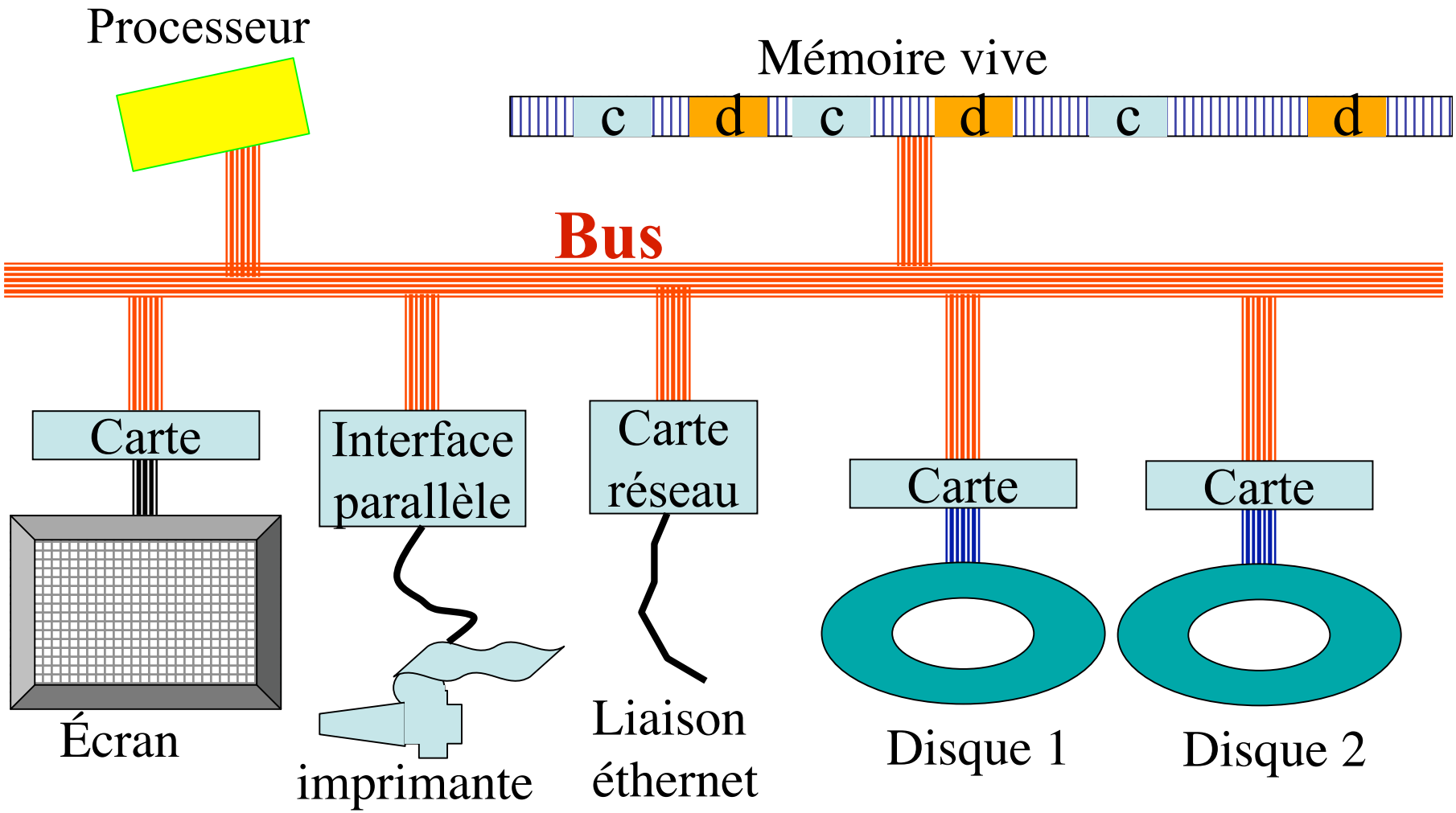
Du matériel au logiciel

Langage, couches, bibliothèques

Le matériel installé



Le rôle des interfaces

- Ce sont de petits systèmes informatiques
- Le processeur communique avec elles en utilisant pour chacune ses adresses particulières
- Il leur envoie des commandes et envoie ou reçoit des données

Comment peut-on programmer ??

- Faut-il connaître le matériel ?

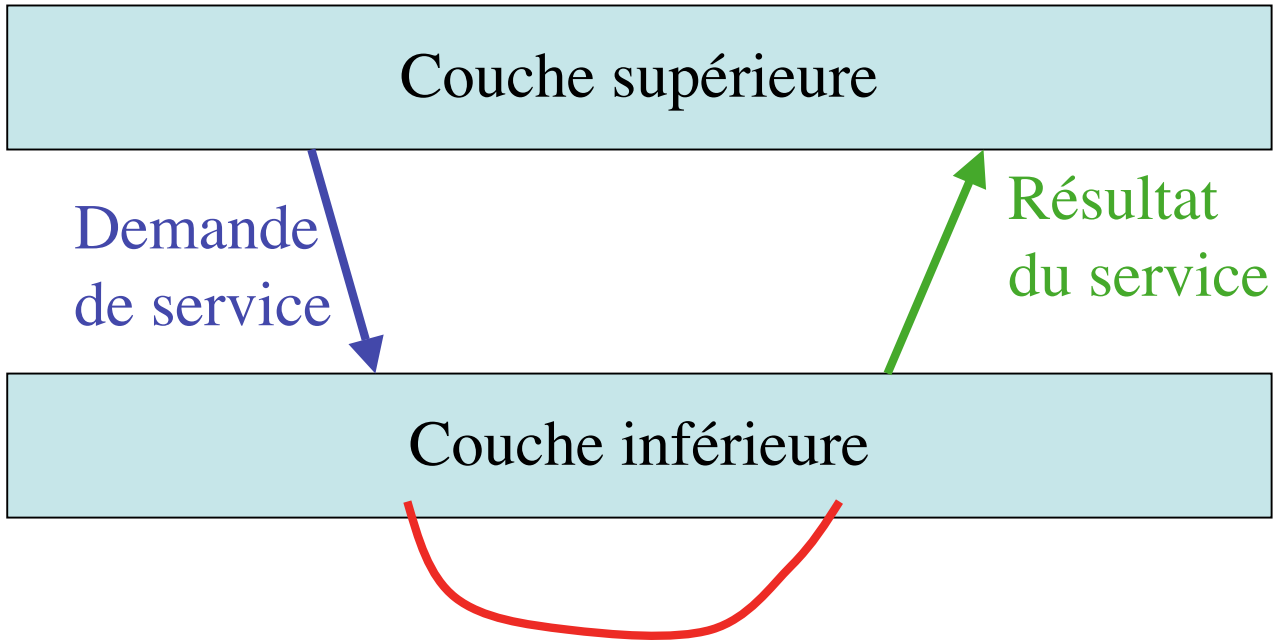
non

- C'est le rôle du système d'exploitation
- Une idée fondamentale à comprendre :

le logiciel est organisé en couches

Qu'est-ce qu'une couche logicielle ?

On écrit un programme (la couche inférieure) qui fonctionne comme un **prestataire de services** pour le compte d'autres programmes (la couche supérieure)


Couche supérieure
Demande
de service
Résultat
du service
Couche inférieure
Exécution de la couche inférieure

Avantage

- Sépare les problèmes
- Chaque fois qu'on demande le même service, on sait qu'on appelle le même code
- La couche inférieure peut avoir été écrite avant, par d'autres programmeurs (et être déjà déboguée)

Comment fait-on

- Il faut une technique pour faire les demandes de service et transmettre les réponses
  - La plus simple : demande = appel de fonction, réponse = valeur de retour
- Il faut fixer :
  - La liste des demandes de services possibles
  - La signification de la réponse dans tous les cas de figure

- L'ensemble (technique de demande de service + liste des demandes + spécification de leur réponse) s'appelle une **interface de programmation** (API)
- Idée de base : quand on connaît l'API, on n'a pas besoin de savoir comment la couche est programmée

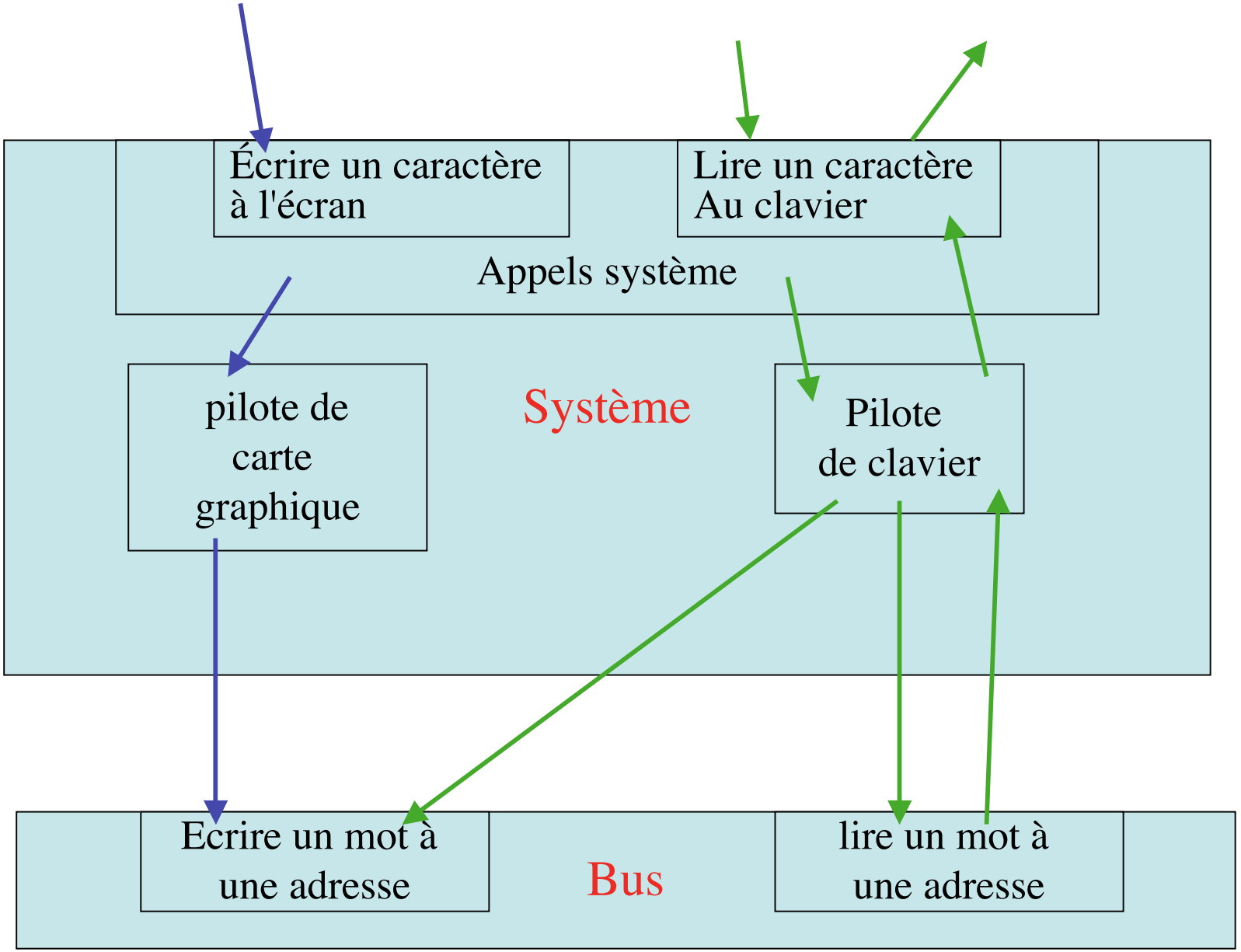
Un exemple

- Dans une couche haute du système, il y a des bibliothèques, par exemple la bibliothèque standard **libc**
- Trois exemples de demandes de services à la **libc** :
  - printf() : calcule une chaîne de caractères, l'écrit à l'écran à partir de la position du curseur, renvoie le nombre de caractères écrits

- getchar() : attend qu'une touche soit reçue du clavier. Renvoie son code ascii en tant qu'entier.
- scanf() : lit une chaîne de caractère au clavier en pratiquant au passage des conversions dont le résultat est stocké dans les variables indiquées. Renvoie le nombre de conversions réussies.
- **getchar**(), **printf**(), **scanf**() et leur définition font partie de l'API de la bibliothèque standard

Le rôle du système

- Le système d'exploitation est (en simplifiant) une couche qui assure l'accès au matériel.
- Les demandes de service au système sont des **appels systèmes**
- Ces appels utilisent des modules logiciels dépendant du matériel qu'on appelle des **pilotes** (*drivers* en anglais)


Écrire un caractère
à l'écran
Lire un caractère
Au clavier
Appels système
pilote de
carte
graphique
Système
Pilote
de clavier
Ecrire un mot à
une adresse
Bus
lire un mot à
une adresse

La portabilité

- Quand on change de machine, on peut avoir un autre modèle de clavier, de carte graphique, de disque.
- donc les pilotes ne sont pas les mêmes.
- Et si on change de modèle de processeur…
- le langage machine change aussi.

Langage évolué 1.

Un langage évolué, c'est un double contrat :

1. Premier contrat : ce que le langage comprend et comment il le comprend. Ex :
   - En C, on écrit **x += 3 ;** en Pascal, il faut écrire **x := x + 3 ;**
   - En C, on écrit **m = milieu(a,b) ;** en Java, il faut écrire **m = a.milieu(b) ;**

Explication : le travail du compilateur

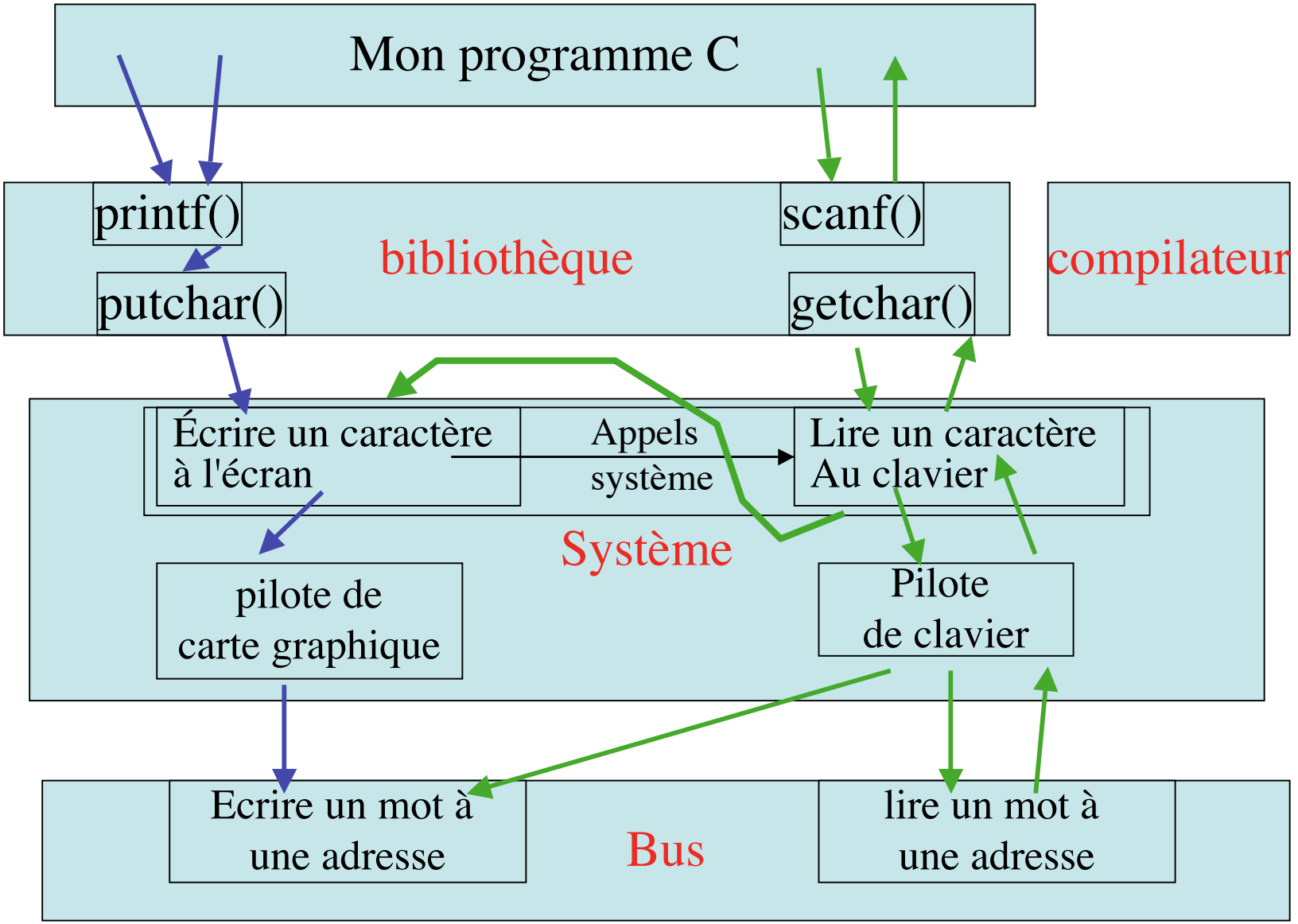
- Assembleur : langage proche de la machine
  - il faut connaître le processeur pour s'en servir
  - il y a des noms pour les adresses mémoire et les instructions (ex : add x 3)
  - le fabriquant du processeur définit l'assembleur
  - Il se traduit directement en langage machine
- Langage évolué : un langage que le compilateur va traduire **en assembleur**

"le langage comprend"

- "le compilateur comprend" = il réussit à traduire en assembleur (pas d'erreur de compilation).
- En fait, un compilateur ne sait traduire que
  - des instructions arithmétiques : **+**, **-** , **>**, **!=**, …
  - des commandes d'exécution : **if**, **while**, …
  - des appels de fonction **x = f(y,z)** à condition de savoir où est la fonction

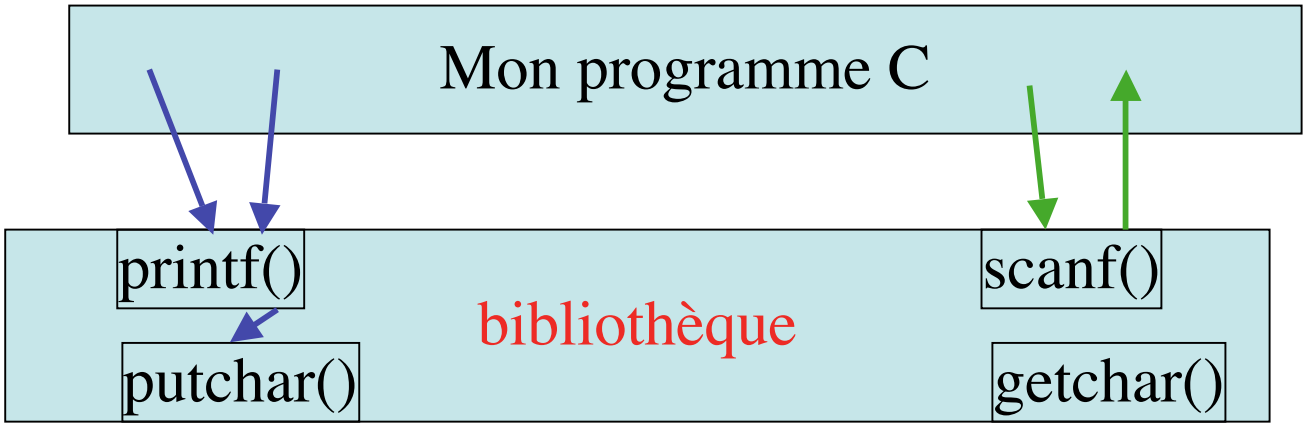
Langage évolué 2.

2. Le deuxième élément du contrat, ce sont les bibliothèques.
   - tous les compilateurs C sur toutes les machines sont fournis avec une bibliothèque standard qui contient une version de printf(), scanf(), getchar(), etc.
   - Il y a beaucoup d'autres bibliothèques plus ou moins spécialisées : pour accéder au réseau, pour tirer un nombre au hasard, …


Mon programme C
printf()
putchar()
bibliothèque
scanf()
getchar()
compilateur
Écrire un caractère
à l'écran
Appels
système
Lire un caractère
Au clavier
Système
pilote de
carte graphique
Pilote
de clavier
Ecrire un mot à
une adresse
Bus
lire un mot à
une adresse

A quoi ça sert ?

- En théorie, on n'a besoin que de connaître le langage :



En réalité

- Il y a des différences entre compilateurs

  Ex. : pour certains, unsigned int va de 0 à 65535. Pour d'autres, c'est de 0 à 4294967296

- Il y a des différences entre appels de fonction.

  Ex. : scanf() n'est pas le même sous unix et sous windows (parce que l'appel système n'est pas le même)

- Il y a des problèmes d'installation

  Ex : sur cette machine, la bibliothèque de tirage au sort n'est pas fournie (ou : elle porte un autre nom)

  Ex : le système est trop vieux et il n'y a pas de pilote pour ce type de disque

  Etc., etc.

- **Il faut savoir comprendre où est le problème !!!**

La forme la plus simple de bibliothèque

- On écrit des fonctions dans un fichier C. Ex

```
/* fichier : formes.c */
int ligne(int dir, int long){…}
int rect(int long, int larg){…}
```

- On compile le fichier en objet :

```
cc -c formes.c -o formes.o
```

Option -c : on veut fabriquer un fichier objet

Le fichier source

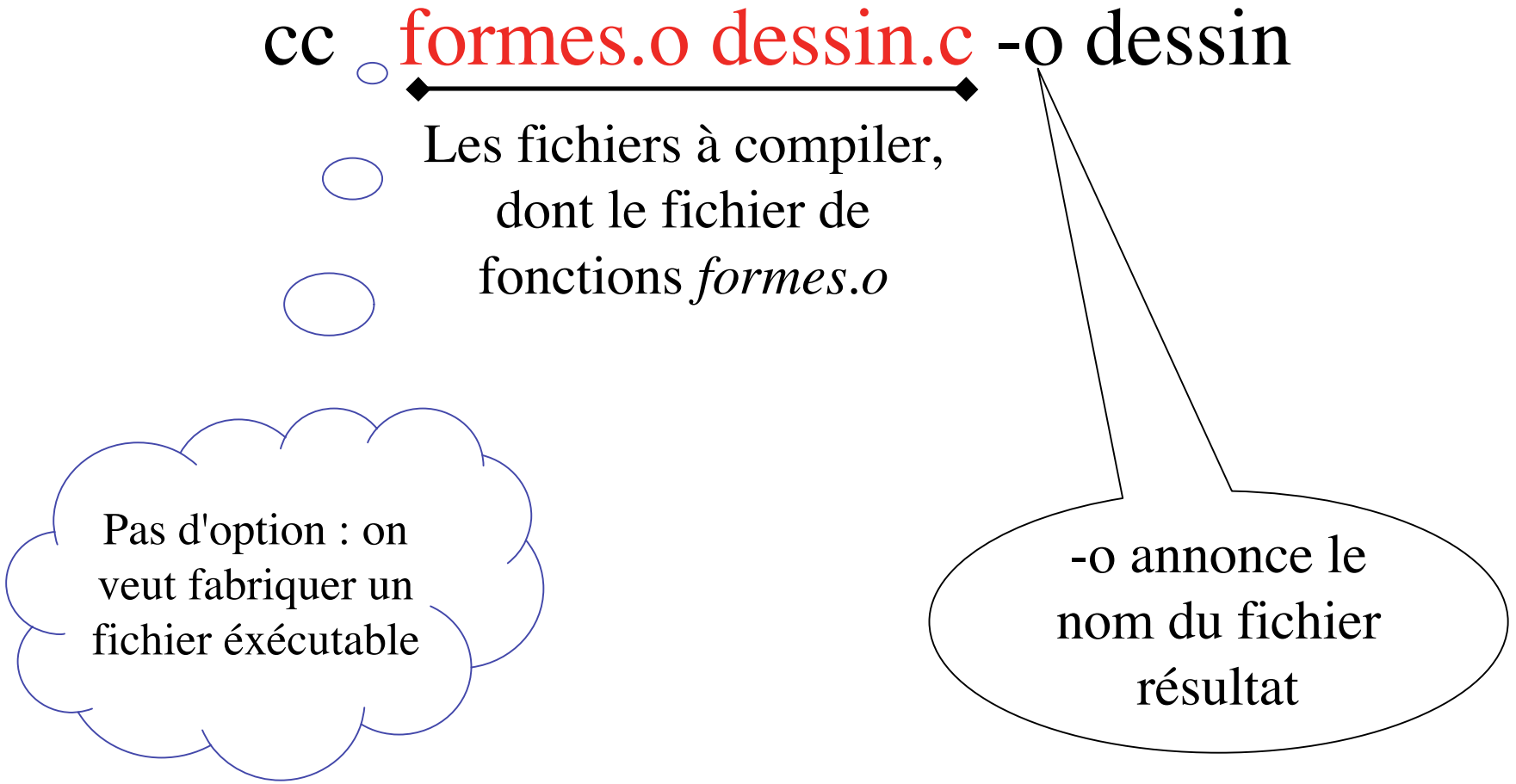
-o annonce le nom du fichier résultat

On utilise ces fonctions dans un autre fichier source. Ex. :

```
/* Fichier : dessin.c */
int ligne(int, int);
int rect(int, int) ;
int main(void) {
    ligne(15, 80) ;
    rect(40,60) ;
    return (0) ;
}
```

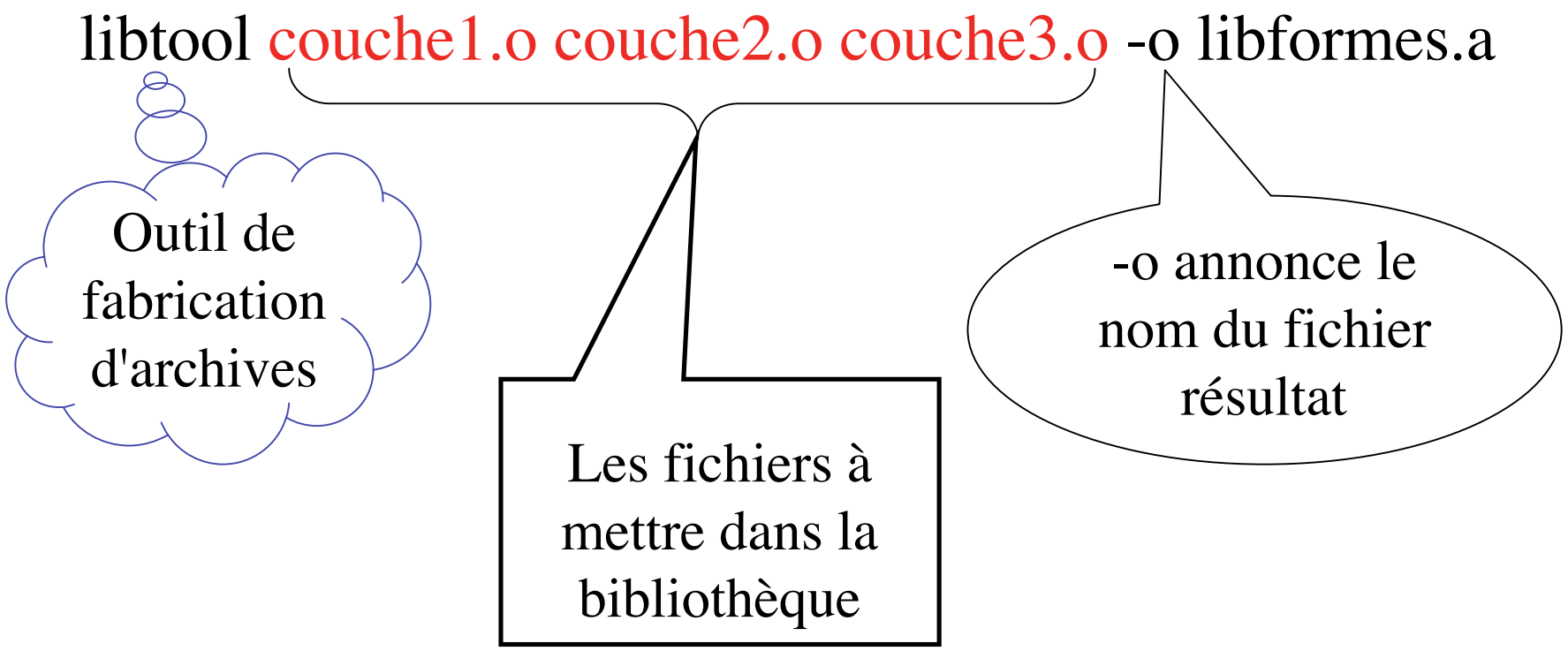
Comme ligne() et rect() sont dans un autre fichier, il faut re-donner leur type

On fait le lien **à la compilation** :



Bibliothèque des fichiers objet archivés

- On fait une archive



- On a deux façons de compiler

gcc dessin.c libformes.a -o dessin

La bibliothèque est **à la fin** de la liste des fichiers à compiler

gcc dessin.c -L. -lformes -o dessin

-L indique dans quel répertoire chercher la bibliothèque si elle n'est pas dans /usr/lib

-l indique le nom de la bibliothèque