المعهد العالي للدراسات التكنولوجية في المواصلات بتونس
Institut Supérieur des Etudes Technologiques en Communications de Tunis
Iset'Com

Jihen Thabet

Module:
Informatique embarquée: Programmation C embarquée

2017/2018

Plan

1. Objectifs
2. Introduction
3. Les principes de base de la compilation
4. Remarques générales
5. Eléments de base
6. Opérateurs et expressions
7. Structures de contrôle de programme
8. Visibilté des données
9. Pointeurs
10. Les entrées sorties
11. Programmation C avancée
12. Les bibliothèques standards
13. Exemples de programmation C pour l'embarqué
14. Quelques pièges classiques
15. Les threads en C

Objectifs

- Balayer et revoir les aspects **importants et essentiels** du langage C que doit maîtriser tout ingénieur électronicien afin de concevoir le logiciel de base d'un système numérique (système embarqué).
- Connaître les points forts et les points faibles du langage C afin d'éviter les pièges classiques.
- Maîtriser les appels d'E/S de base et formatés en langage C.
- Comprendre comment on développe une application embarquée en langage C à travers des exemples.

Introduction

- Aucun langage de programmation n'a pu se vanter d'une croissance en popularité comparable à celle de C.
- Le langage C n'est pas un nouveau né dans le monde informatique :
  - 1972 : développement par ATT Bell Laboratories.
- But :
  - Développer une version portable d'un système d'exploitation ouvert qui ne plante pas afin de mettre en commun les imprimantes des secrétaires du laboratoire.

Avantages du langage C

- C est un langage :
  - Universel : C n'est pas orienté vers un domaine d'applications spéciales comme SQL
  - Compact : C est basé sur un noyau de fonctions et d'opérateurs limités qui permet la formulation d'expressions simples mais efficaces.
  - Moderne : C est un langage structuré, déclaratif et récursif. Il offre des structures de contrôle et de déclaration comme le langage Pascal.
  - Près de la machine : C offre des opérateurs qui sont très proches de ceux du langage machine (manipulations de bits, pointeurs…). C'est un atout essentiel pour la programmation des systèmes embarqués.

Avantages du langage C

- Rapide : C permet de développer des programmes concis et rapides.
- indépendant de la machine :C est un langage près de la machine (microprocesseur) mais il peut être utilisé sur n'importe quel système ayant un compilateur C.
- Portable : En respectant le standard ANSI-C, il est possible d'utiliser (théoriquement ☺) le même programme sur tout autre système (autre hardware, autre système d'exploitation),simplement en le recompilant. Il convient néanmoins de faire attention dans le cas d'un système embarqué qui n'inclut pas toujours un système d'exploitation
- Extensible : C peut être étendu et enrichi par l'utilisation de bibliothèque de fonctions achetées ou récupérées (logiciels libres…).

Inconvénients du langage C

- Efficacité et compréhensibilité : C autorise d'utiliser des expressions compactes et efficaces. Les programmes sources doivent rester compréhensibles pour nous-mêmes et pour les autres.

Inconvénients du langage C

- Exemple :

Les deux lignes suivantes impriment les N premiers éléments d'un tableau A[] en insérant un espace entre les éléments et en commençant une nouvelle ligne après chaque dixième chiffre :

```
for (i=0; i<n; i++)
        printf("%6d%c", a[i], (i%10==9)?'\n':' ');
```

Cette notation est très pratique mais plutôt intimidante pour un débutant. L'autre variante est plus lisible :

```
for (I=0; I<N; I=I+1){
    printf("%6d", A[I]);
    if ((I%10) == 9)
        printf("\n");
    else
        printf(" ");
}
```

Inconvénients du langage C

- portabilité et bibliothèques de fonctions :
  - La portabilité est l'un des avantages les plus importants de C : en écrivant des programmes qui respectent le standard ANSI-C, nous pouvons les utiliser sur n'importe quelle machine possédant un compilateur ANSI-C.
  - Le nombre de fonctions standards ANSI-C (d'E/S) est limité. La portabilité est perdue si l'on utilise une fonction spécifique à la machine de développement.

Inconvénients du langage C

- discipline de programmation :
  - C est un langage qui n'est pas fortement typé comme par exemple Pascal ou Java **(ou VHDL)**.
  - C n'inclut pas de gestion automatique de la mémoire comme Java. **La gestion mémoire en C est la cause de beaucoup de problèmes** (certains ont peut être eu une mauvaise expérience avec les *malloc()/free()*)…
  - L'instruction *goto* existe en C.

LES PRINCIPES DE BASE DE LA COMPILATION

Généralités

- Un langage de programmation a pour finalité de communiquer avec la machine. Il y a diverses manières de communiquer avec la machine.

- Le langage naturel de la machine n'utilise que deux symboles (0 et 1) : c'est le langage machine. Le langage machine ne comprend et n'exécute qu'une suite de 0 et 1 structurée sous forme d'octets (8 bits).

Généralités

- Le langage assembleur permet à l'‘être humain de générer cette suite de 0 et 1 à partir d'un fichier source compréhensible qui utilise des instructions assembleur. Le programme assembleur traduit le code source assembleur en code objet (suite de 0 et 1) exécutable par la machine.
- Le langage C va opérer la même chose que précédemment mais avec encore plus de lisibilité.
- Un programme C est un texte écrit avec un éditeur de texte respectant une certaine syntaxe et stocké sous forme d'un ou plusieurs fichiers (généralement avec l'extension .c).

Généralités

- A l'opposé du langage assembleur, les instructions du langage C sont obligatoirement encapsulées dans des fonctions et il existe une fonction privilégiée appelée *main()* qui est le point d'entrée de tout programme C (sous UNIX).
- Dans le cas d'un système embarqué sans système d'exploitation, la fonction spécifique main() perd son sens. Le point d'entrée du programme sera celui spécifié (en mémoire ROM) dans la table des vecteurs pour l'initialisation du compteur programme au RESET

Programme C : Hello world

- Fichier source C hello.c :

```
#include <stdio.h>
int main() {
        printf("Hello world \n");
        return(0) ;
    }
```

Affichage de "Hello world " à l'écran.

Programme C : Hello world

- Pour afficher cette chaîne, le programme fait appel à la fonction printf qui fait partie des appels d'E/S standards définis par la norme C-ANSI.
- L'ensemble de ces fonctions standards (appelé bibliothèque C) est stocké dans un ou plusieurs fichiers (ici libc.a).

Compilation et édition de liens

- La traduction du fichier texte ci-dessus en un programme exécutable se décompose en deux phases :

1. la compilation qui est la traduction d'un programme C en une suite d'instructions machine. Le résultat produit est un fichier objet (généralement avec l'extension .o).
2. l'édition de liens produit à partir d'un ou de plusieurs fichiers objets et des bibliothèques un fichier exécutable.

- Outre la compilation des divers fichiers objets, l'édition des liens inclut le code objet des fonctions standards utilisées par le programme.

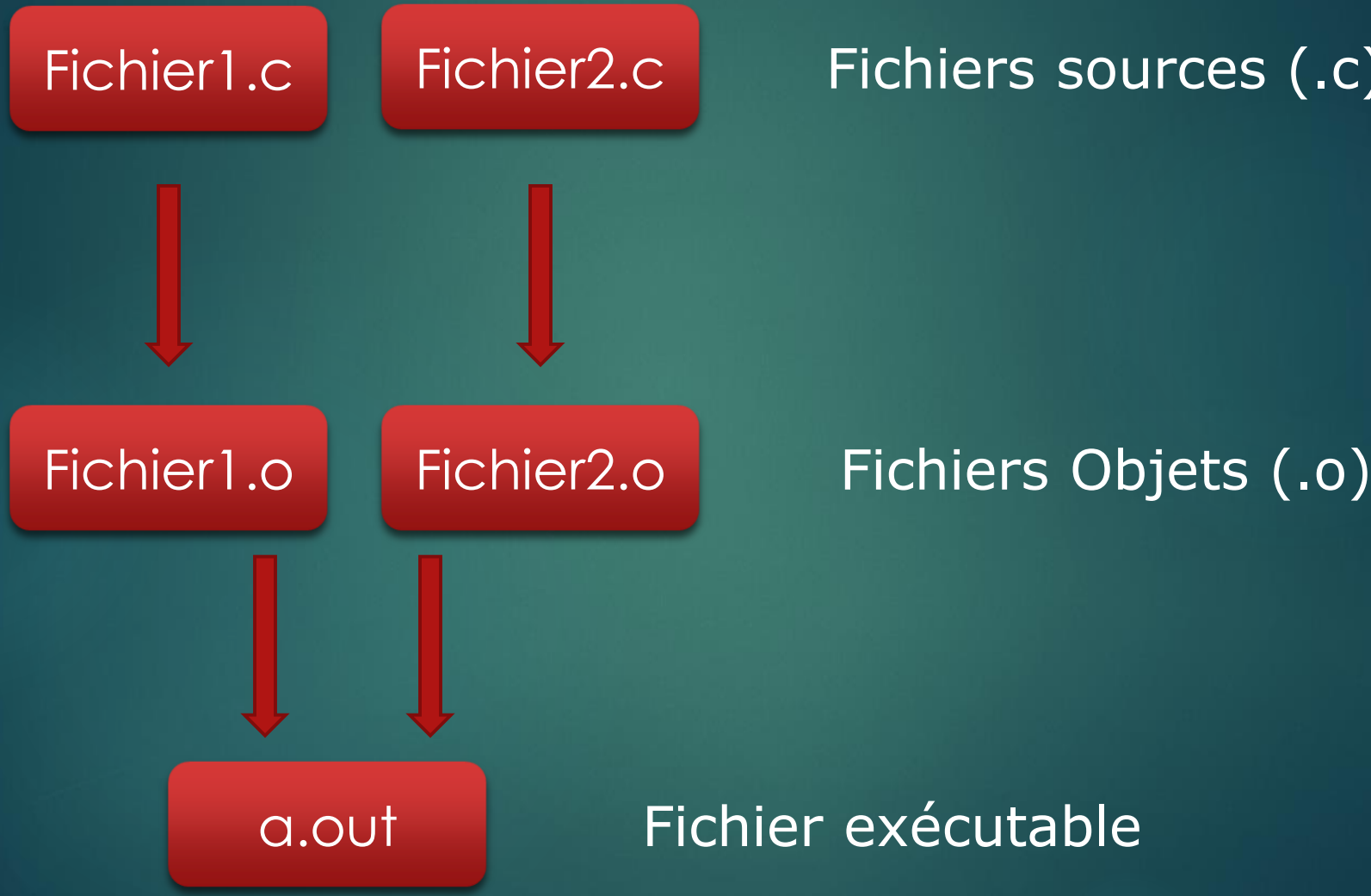
Compilation et édition de liens

- Les compilateurs C font subir deux transformations aux fichiers sources :

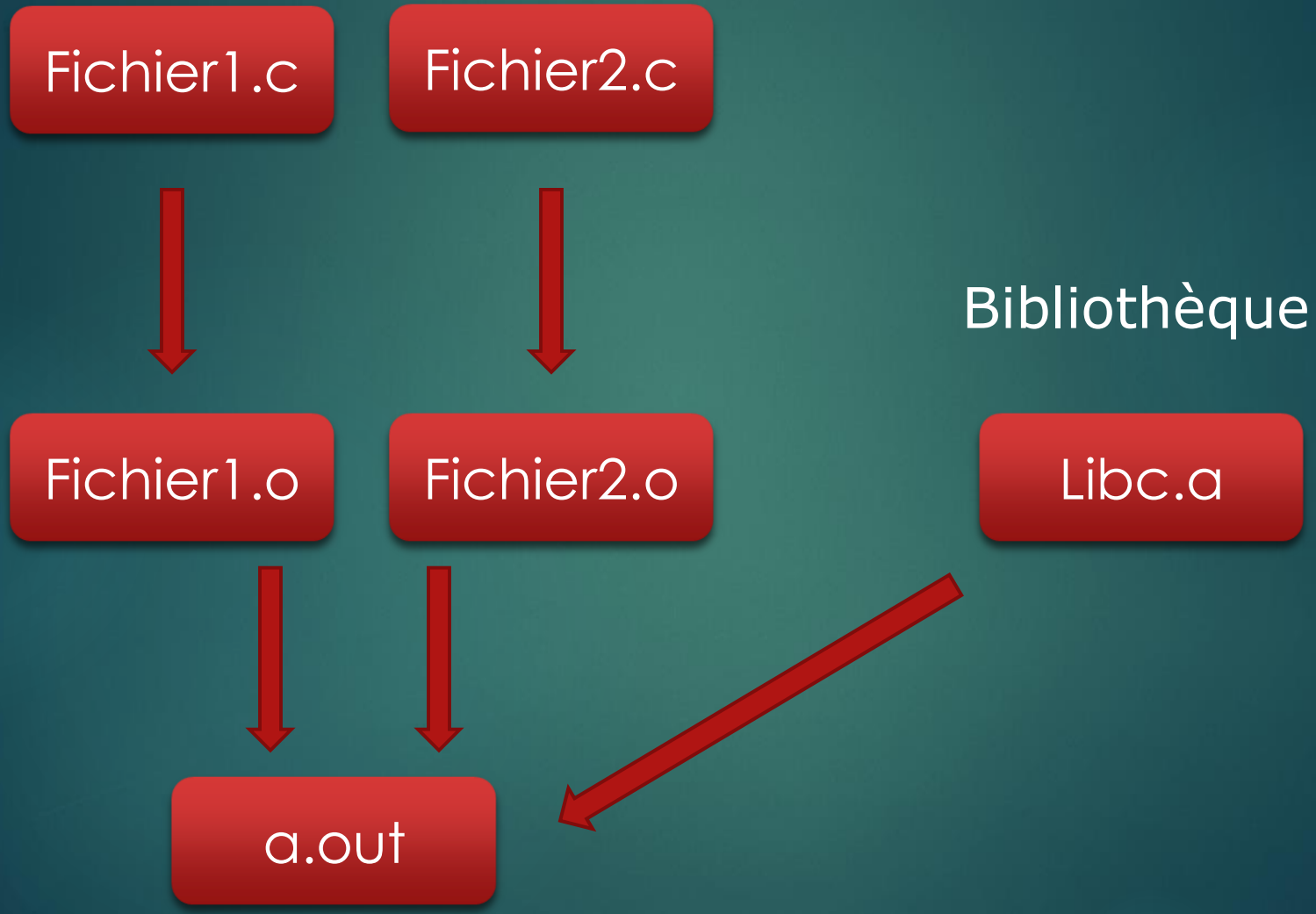
1. un préprocesseur réalise des transformations d'ordre purement textuel pour rendre des services du type inclusion de source, compilation conditionnelle et traitement de macros.
2. le compilateur C proprement dit prend le texte généré par le préprocesseur et le traduit en code objet.

- La fonction du préprocesseur est assez souvent implémentée par un programme séparé (cpp sous Linux) qui est automatiquement appelé par le compilateur C.

Compilation et édition de liens



Compilation et édition de liens



Compilation et édition de liens

- Chaque fichier source (ou texte) est appelé *module* et est composé :
- ✓ des définitions de fonctions
- ✓ des définitions de variables
- ✓ des déclarations de variables et des fonctions externes
- ✓ des directives pour le préprocesseur (lignes commençant par #) de définitions de types ou de constantes (struct, union...).

- **C'est la base de la programmation modulaire !**

Fichiers « include »

- Pour compiler correctement un fichier, le compilateur a besoin d'informations concernant les déclarations des structures de données et de variables externes ainsi que de l'aspect (*prototype*) des fonctions prédéfinies.
- Un programme se compose donc :

✓ De structures de données.

✓ D'algorithmes utilisant ces structures de données.

Fichiers « include »

- Ces informations sont contenues dans des fichiers *header* avec l'extension .h. Ces fichiers doivent être inclus dans le fichier source que l'on veut compiler.
- Pour cela, le langage C offre la directive du préprocesseur :

```
#include nom_de_fichier
```

Fichiers « include »

- Exemple pour utiliser l'appel printf (affichage à l'écran) :

```
#include <stdio.h>
```

- stdio.h : contient les déclarations de variables externes et les prototypes de fonctions de la bibliothèque d'E/S formatées (*standard input output*).

Les commentaires

- Dès que l'on écrit un programme important, il est indispensable d'y inclure des commentaires qui ont pour but d'expliquer ce qu'est sensé faire le programme rendant le programme lisible à soi même et à autrui ☺.
- **Il est important d'assurer la maintenance du code ; ce qui est bien programmer !**
- Un commentaire commence par les caractères /* et se termine par */.
- L'erreur classique avec les commentaires est d'oublier la séquence fermante */.
- Dans ce cas, le compilateur va considérer que le commentaire se poursuit jusqu'à la fin du prochain commentaire. Ceci peut ne pas générer d'erreur syntaxique.

ELEMENTS DE BASE

Les types de données élémentaires

- Un programme C manipule deux types de données de base :
- ✓ les nombres entiers.
- ✓ les nombres flottants.
- Un autre type de données élémentaires est le *pointeur* (adresse d'une variable).
- Toute autre structure de données sera dérivée à partir de ces types fondamentaux.
- Par exemple, les caractères, les booléens, les constantes ... ne sont rien d'autres que des nombres.

Les types de données élémentaires

- Notation binaire, décimale, hexadécimale, octal:
- Rappelons qu'un nombre *n* en notation décimale est représenté en base *b* par le nombre:

$a_i\ a_{i-1}\ a_{i-2}\ \ldots\ldots\ a_1\ a_0$

$n = a_m * b^m + a_{m-1} * b^{m-1} + \ldots + a_1 * b^1 + a_0\ b^0$ avec $0 <= a_i < b$.

- Voici une liste des bases habituellement utilisées en (micro)informatique :

Base	Notation	Symbols
2	Binaire	0,1
8	Octale	0,1…,7
10	Décimale	0,1…,9
16	Héxadécimale	0,1…,9,a,b,c,d,e,f

Les types de données élémentaires

- Exemple :

Le nombre 70 (en notation décimale) est représenté par 1000110 en notation binaire, 0106 en notation octale, 0x46 en notation hexadécimale.

Les types de données élémentaires

- Représentation des nombres signés (positifs et négatifs)
- ✓ On appelle bit de signe, le bit le plus élevé (*bit de poids fort ou MSB Most Significant Bit*) d'un nombre entier.
- ✓ Si l'on considère un nombre codé sur un octet, lorsque le bit de signe vaut 0, il s'agit d'un nombre positif. Inversement lorsque ce bit vaut 1, il s'agit d'un nombre négatif.

Les types de données élémentaires

- Exemple sur 8 bits :

  01111111 : +127

  10000000 : -128

  11111111 : -1

  00000000 : 0 (+0, -0)

- Ce format de représentation d'un entier est **appelé complémentation à 2** qui permet d'avoir une seule unité arithmétique et logique (ALU) pour manipuler des nombres entiers positifs ou négatifs, une soustraction se ramenant à une addition et assure l'unicité de représentation du nombre 0.

Les types de données élémentaires

- Entiers non signés :

- ✓ Dans un octet, il est possible de ranger 2^8 valeurs différentes. Si l'on décide que cet octet est susceptible de contenir des entiers positifs et négatifs (le bit de signe est occupé), on codera les valeurs comprises entre -2^7. et 2^7-1. Inversement, si l'on décide que cet octet ne contient que des entiers non signés (positifs ou nuls), on codera les valeurs comprises entre 0 et 2^8-1

- ✓ **C'est à nous de savoir à priori si l'on travaille sur un nombre entier signé ou non signé (structure de données) et non le processeur**

Les types de données élémentaires

- Le langage C considère par défaut les variables de base comme signées.
- Dans les structures de données utilisées par un programme, il faut se poser la question du type signé ou non.
- Si l'on travaille en valeur absolue (non signé), il est obligatoire d'ajouter la qualificatif *unsigned* sous peine d'effets de bord en cours d'exécution !

Les entiers

- En C, on dispose de divers types d'entiers qui se distinguent par la place qu'ils occupent en mémoire :

sur 1 octet, les entiers *signés* et *non signés* (char) et (unsigned char).

sur 2 octets, les entiers *signés* et *non signés* (short) et (unsigned short).

sur 4 octets (**en général**), les entiers *signés* et *non signés* (long) et (unsigned long).

le type int (unsigned int) est selon les machines synonyme de short (unsigned short) ou de long (unsigned long).

- Il faudra à priori s'assurer de la taille de représentation du type *int* sur sa machine en utilisant par exemple l'instruction sizeof :

```
printf("int=%d octets\n", sizeof(int));
```

Le type char

- Le type char désigne un nombre **entier signé codé sur 1 octet**.

Code ASCII

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Le type char

- Toutes les opérations autorisées sur les entiers peuvent être utilisées sur les caractères :

on peut ajouter ou soustraire deux caractères, ajouter ou soustraire un entier à un caractère ☺.

- Exemple :

Conversion du caractère c désignant un chiffre en sa valeur numérique v correspondante :

```
v = c - '0';
```

Les types short, long ou int

- Le type short représente un entier signé codé sur 2 octets (de -32768 à 32767) et le type unsigned short représente un entier non signé codé sur 2 octets (de 0 à 65535).
- Le type long (ou int suivant la machine) représente un entier signé codé sur 4 octets (de -2147843648 à 2147843647) et le type unsigned long (ou unsigned int) représente un entier non signé codé sur 4 octets (de 0 à 4294967295).

Le type réel

- Les nombres à *virgule flottante* (abusivement appelés réels) servent à coder de manière approchée les nombres réels. Un nombre à virgule flottante est composée d'un signe, d'une mantisse et d'un exposant.
- On dispose de trois types de nombres à virgule flottante :

- ✓ float
- ✓ double
- ✓ long double

Le type réel

- Les floats :

Un float est codé sur 4 octets avec 1 bit de signe, 23 bits de mantisse et 8 bits d'exposant (valeurs comprises entre $-3.4 * 10^{-38}$ et $+3.4 * 10^{38}$).

- Les doubles :

Un double est codé sur 8 octets avec 1 bit de signe, 52 bits de mantisse et 11 bits d'exposant (valeurs comprises entre $-1.7 * 10^{-308}$ et $+1.7 * 10^{308}$).

- Les long doubles :

Un long double est codé sur 10 octets avec 1 bit de signe, 64 bits de mantisse et 15 bits d'exposant (valeurs comprises entre $-3.4 * 10^{-4932}$ et $+3.4 * 10^{4932}$).

Les constantes littérales

- Les constantes de type caractère:

Les constantes de type caractère se notent entre apostrophes ' : 'a' '2' ""‘

Le caractère ' se note '\'' et le caractère \ se note '\\'. On peut également représenter des caractères non imprimables à l'aide de *séquences d'échappement :*

Séquences d’échappement

\a	alerte (avertisseur sonore)
\b	retour arrière (backspace)
\c	suppression du retour-chariot final
\f	saut de page
\n	nouvelle ligne
\r	retour-chariot
\t	tabulation horizontale
\v	tabulation verticale
\\	backslash
\nnn	le caractère dont le code ASCII octal vaut nnn (un à trois chiffres)
\xnnn	le caractère dont le code ASCII hexadécimal vaut nnn (un à trois chiffres)

Les chaînes de caractères

- Les chaînes de caractères se notent entre guillemets :

"coucou"

"C'est bientot fini !!!"

- Une chaîne de caractères est une suite de caractères (éventuellement vide) entre guillemets. Il en découle que l'on est autorisé à utiliser les séquences d'échappement dans les chaînes.
- En mémoire, une chaîne de caractères est une suite de caractères consécutifs et dont le dernier élément est le caractère nul '\0'.

Les chaînes de caractères

- Une chaîne de caractère doit être écrite sur une seule ligne. Lorsqu'il est trop long pour tenir une même ligne, on découpe celle-ci en plusieurs bouts. Chaque bout étant écrite sur une seule ligne et on masque le retour à la ligne par le caractère \.

Les booléens

- Contrairement à d'autres langages (comme Pascal), il n'y a pas de type booléen en C.
- Le type booléen est représenté par un entier. Il se comporte comme la valeur booléenne *vraie* si cette valeur entière est non nulle.
- Dans un contexte qui exige une valeur booléenne (comme les tests, par exemple), un entier non nul équivaut à *vrai* et la valeur nulle équivaut à *faux*.
- De même, une fonction qui retourne une valeur booléenne pourra retourner une valeur non nulle comme équivalent à *vrai* et la valeur 0 comme équivalent à *faux*.

Les variables

- Une variable possède :

Un nom (un *identificateur*) composé d'une suite de caractères commençant par un caractère alphabétique et suivi de caractères alphanumériques ou du caractère _.

- Exemple :

x, x1, x_1, Var, VarLocal, var_local

- Contrairement aux constantes, le *contenu* d'une variable peut être modifiée à volonté
- ce qui ne change pas c'est l'adresse de la variable.

Les variables

- Définition et déclaration de variables :
- ✓ En C, toute variable utilisée dans un programme doit auparavant être définie.
- ✓ La *définition* d'une variable consiste à la nommer et lui donner un type et éventuellement lui donner une valeur initiale. C'est cette définition qui réserve (*alloue*) en fonction de son type la place mémoire nécessaire.
- ✓ Initialiser une variable consiste à remplir, avec une constante, la zone mémoire réservée à cette variable. Cette opération s'effectue avec l'opérateur =
- Il ne faut pas confondre l'initialisation et l'affectation. Malheureusement, ces deux opérations utilisent le même symbole =.

Les variables

- Exemple :

```
int x = 2;
char c = 'c';
float f = 1.3;
```

- Lorsque le programme que l'on réalise est décomposé en plusieurs modules, une même variable, utilisée dans plusieurs modules, doit être déclarée dans chacun de ces modules. Par contre, on ne définira cette variable que dans un seul de ces modules.
- C'est au moment de l'édition des liens que l'on mettra en correspondance les variables apparaissant dans plusieurs modules (variable partagée, mot clé *extern*).

Les variables

- Exemple :

```
int x, y = 0, z;
extern float a, b;
unsigned short cpt = 1000;
```

- Portée des variables :
- ✓ La position de la déclaration ou de la définition d'une variable détermine sa portée, sa durée de vie et sa visibilité.

Les variables *globales* sont déclarées en dehors de toute fonction.

Les variables *locales* sont déclarées à l'intérieur des fonctions et ne sont pas visibles à l'extérieur de la fonction dans laquelle celle-ci est définie.