Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2

a.02 - Introduction au C

Classes T-2/I-2 // 2018-2019

Daniel Gachet | HEIA-FR/TIC
a.02 | 10.09.2018

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

Contenu

- Introduction
- Historique
- Caractéristique du langage
- Outils
- Syntaxe
- Librairies standard

Qu'est ce qu'un programme?

- **Ensemble des activités liées à la définition, l'écriture, la mise au point et l'exécution de programmes informatiques ; séquence des ordres auxquels doit obéir un dispositif.**

  Dictionnaire Larousse en ligne (http://www.larousse.fr/dictionnaires/francais/interruption)

- **Un programme informatique est un algorithme, lequel est constitué de 2 éléments de base:**
  1. Les **actions**, lesquelles sont décrites par une suite d'**instructions** devant être exécutées
  2. Les **données**, lesquelles sont représentées par des **variables** manipulées durant l'exécution des instructions

Contenu du cours C

- **1ère étape**
  - Introduction
  - Eléments de base du langage
  - Structures complexes
  - Pointeurs et mémoire dynamique
  - Pointeurs de fonction
- **2ème étape**
  - Fonctions et interfaçage avec l'assembleur
- **3ème étape**
  - Design OO
- **4ème étape**
  - Chaîne d'outils de développement
  - Tests unitaires et couverture du code
  - Documentation

Histoire du C

~1970 Dennis Ritchie et Kenneth Thompson décident d'apporter des améliorations au langage B et de créer un nouveau B (NB)

1972 Les améliorations du langage sont telles que Ritchie décide de baptiser ce nouveau langage C

1973 90% de UNIX est écrit en C

1974 Les laboratoires Bell accordent des licences UNIX aux universités et popularisent ainsi le langage de programmation C

1978 Brian Kernighan et Dennis Ritchie formalisent le langage, appelé aujourd'hui K&R C, et éditent le premier ouvrage « The C Programming Language »

1983 Bjarne Stroustrup ajoute, dès le début des année 80, le concept de classe à C et crée en 1983 C++

1989 Standardisation du langage et création du ANSI C (C89)

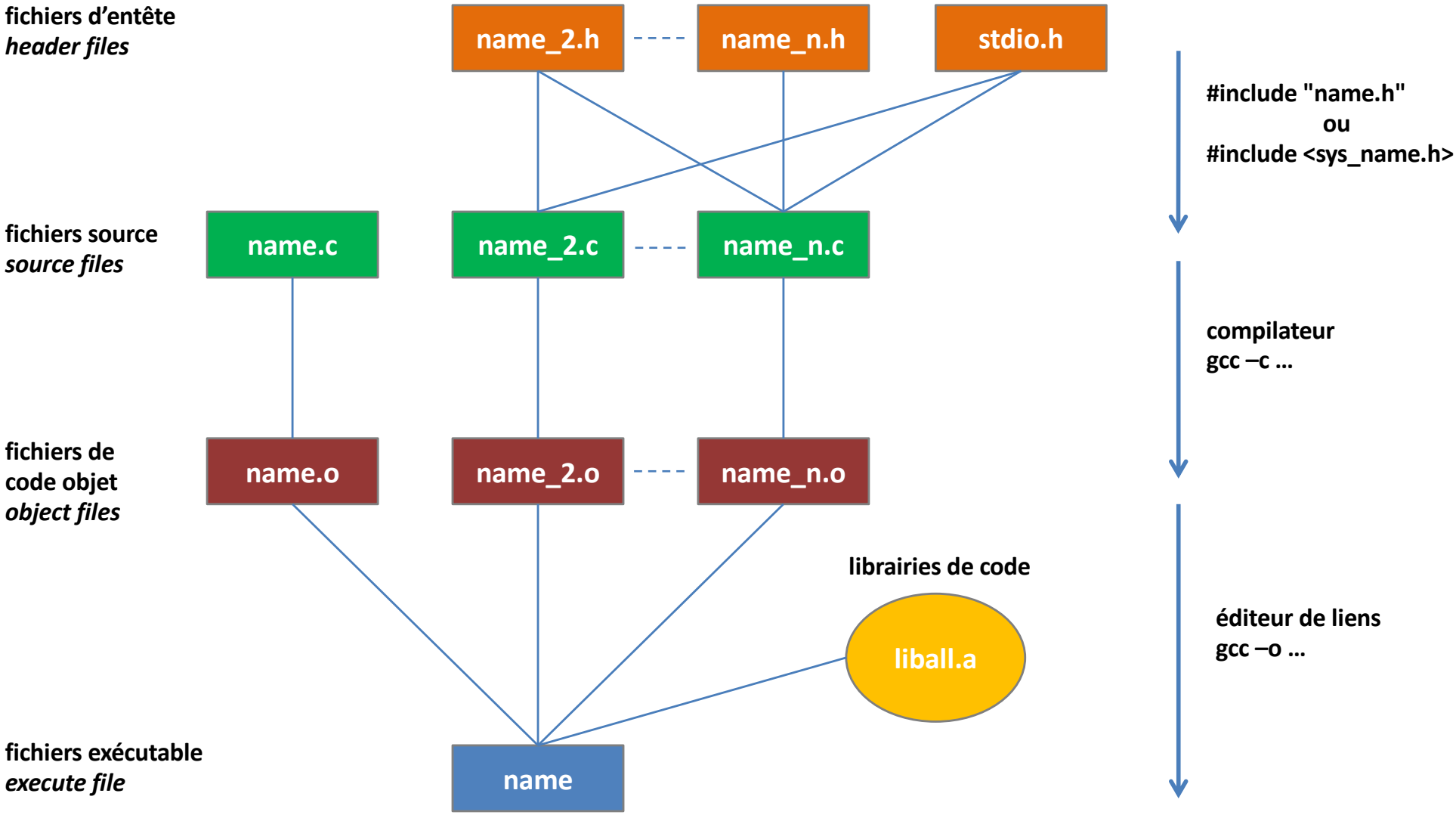
1999 Révision du langage (C99)

2011 Dernière évolution du langage (C11), anciennement (C1X)

Caractéristiques...

- **Langage pas de très haut niveau, mais offrant une programmation ouverte permettant la manipulation d'objets élémentaires (bits, octets, ...) et un accès au matériel.**
- **Le langage ne possède pas d'instruction d'entrée/sortie, de gestion de la mémoire, de système de gestion de communication et de processus. Tous ces mécanismes sont fournis par le système à l'aide de bibliothèques de fonctions et d'appels aux systèmes d'exploitation.**
- **Puissance du langage:**
  - Vitesse d'exécution (compilation optimale)
  - Opérateurs puissants pour les nombres entiers
  - Utilisation généralisée des pointeurs
  - Accès facilité au matériel
- **Normalisation du langage:**
  - Kernighan & Ritchie C, version primitive du langage
  - ANSI-C, standardisation du prototypage, normalisation des librairies standard
  - POSIX, normalisation des fonctionnalités relevant des systèmes d'exploitation

Organisation des fichiers sous C



- **Sous Linux, la chaîne est installée par défaut**

- **Sous Mac OS X**
  - Installer la « command line tools » à l'aide de Xcode

- **Sous Windows:**
  - MinGW - Chaine d'outils GNU pour Windows:
    - http://www.mingw.org/
  - GDB – GNU debugger pour Windows:
    - http://sourceforge.net/projects/mingw/files/GNU Source-Level Debugger/

- **Environnement de développement**
  - Eclipse CDT:
    - http://www.eclipse.org/

Les programmes C ont généralement la structure suivante

```
/**
   inclusion (#include <header.h>) des fichiers de déclaration (header files)
   pour les fonctions, macros ou variables externes définies dans d'autres
   fichiers source.
*/

// définition des types de données
// définition des variables globales
// définition des fonctions locales et globales

int main ()
{
  // définition des variables locales au bloc
  // instructions implémentant l'algorithme du programme
  return 0;
}
```

La fonction `int main()` constitue le corps de l'application
C'est la première fonction à être appelée lors du lancement d'une application

Commentaires

Tout bon programme doit être documenté. La forme la plus simple et la plus efficace est de documenter directement à l'intérieur du code à l'aide de commentaires.

Tous les compilateurs C connaissent les blocs de commentaires. Ceux-ci sont des séquences de caractères comprise entre `/*` et `*/`, par exemple:

```
/*
 * ceci est une petite ligne de commentaire.
 */
```

Il est important de noter qu'il n'est pas possible d'imbriquer les commentaires. La première occurrence de `*/` termine le commentaire.

Les compilateurs plus récents connaissent également les commentaires de ligne. Ceux-ci débutent par un double slash `//` et se terminent à la fin de la ligne, par exemple:

```
i++; // l'index i est incrémenté de 1
```

Le langage de programmation C est composé de 33 mots clefs (keyword), dans l'ordre alphabétique :

```
auto,
break,
case, char, const, continue,
default, do, double,
else, enum, extern,
float, for,
goto,
if, inline, int,
long,
register, return,
short, signed, sizeof, static, struct, switch,
typedef,
union, unsigned,
void, volatile,
while
```

Le langage propose les directives suivantes

- **Inclusion de fichiers**
  `#include <path-spec>` → fichiers standard
  `#include "path-spec"` → fichiers spécifiques

- **Définition de macros permettant la substitution de texte**
  `#define macro_identifier substution-text`
  `#undef macro_identifier` → pour éliminer la définition d'une macro

- **Définition de fonctions macro**
  `#define macro_fonction_id([argument-list]) substution-text`

- `#define print(x)  printf("x=%d\n", x)`

- **Compilation conditionnelle**

```
#ifdef symbol
#ifndef symbol

#if test expression
     [text-block]
[#elif test expression
     text-block]
[#else
     text-block]
#endif
```

Directives (II)

- **Operateurs**

```
#    → transforme le paramètre d'une macro en un string
##   → crée un token à partir de 2 autres

#define print(x)  printf(#x"=%d\n", x)
#define aa "aa"
#define bb "bb"
#define ab aa ## bb
```

- **Macros prédéfinies**

```
__FILE__     : affiche le nom du fichier source.
__LINE__     : affiche le numéro de la ligne atteinte
__DATE__     : affiche la date de compilation du code source
__TIME__     : affiche l'heure de compilation de la source
__VA_ARGS__  : liste des arguments variables d'une macro (variadic macro)
```

- **Protection pour fichiers d'entête (guard for header files)**

```
#ifndef symbol          /* → par exemple FILENAME_H */
#define symbol

#endif
```

Cette protection basée sur la définition d'un symbole peut être remplacée avantageusement par le pragma «once», mais probablement le plus judicieux est de combiner les deux.

```
#pragma once
```

Exemple

```c
/**
 * A simple exemple of a C program
 */
#include <stdio.h>
#include <stdlib.h>

/**
 * The Fibonacci numbers form a sequence of integers, mathematically defined by
 *   F(0)=0; F(1)=1; F(n) = F(n - 1) + F(n - 2) for n > 1.
 *
 * This results in the following sequence of numbers: 0, 1, 1, 2, 3, 5, 8, 13,
 *    21, 34, 55, 89, 144, 233, 377, 610, 987, ...
 * so that each new number is the sum of the previous two, after seeding the
 * sequence with the starting pair 0, 1.
 *
 * @param n value for which the Fibonacci number should be computed
 * @return Fibonacci number
 */
long fibonacci(long n)
{
  if (n <= 1) return n;
  return fibonacci(n-1)+fibonacci(n-2);
}

/**
 * Main program computing the Fibonacci numbers for a given sequence starting
 * from 0 to a number specified at the command line.
 */
int main (int argc, char** argv)
{
    long n = 0;

    if (argc == 2) {
        n = atol(argv[1]);
    }

    printf("The first %ld Fibonacci numbers are:\n", n);
    printf("%s, %d, %s, %s\n", __FILE__, __LINE__, __DATE__, __TIME__);

    for (long i=0; i<=n; i++) {
        printf("%ld, ", fibonacci(i));
    }
    printf("\n");

    return 0;
}
```

Bibliothèque standard (ANSI-C)

<assert.h>	Contient la macro assert, utilisée pour aider à détecter des incohérences de données et d'autres types de bogues dans les versions de débogage d'un programme.
<ctype.h>	Fonctions utilisées pour classifier rapidement les caractères, ou pour convertir entre majuscules et minuscules de manière indépendante du système de codage des caractères utilisé (character set).
<errno.h>	Ensemble (ou le plus souvent sous-ensemble) des codes d'erreurs renvoyés par les fonctions de la bibliothèque standard au travers de la variable errno.
<float.h>	Contient des constantes qui spécifient les propriétés des nombres en virgule flottante qui dépendent de l'implémentation, telles que la différence minimale entre deux nombres en virgule flottante différents, le nombre maximum de chiffres de précision et l'intervalle des nombres pouvant être représentés.
<limits.h>	Contient des constantes qui spécifient les propriétés des types entiers qui dépendent de l'implémentation, comme les intervalles des nombres pouvant être représentés.
<locale.h>	Pour s'adapter aux différentes conventions culturelles.
<math.h>	Pour calculer des fonctions mathématiques courantes.

Bibliothèque standard (ANSI-C) (II)

<setjmp.h>	Pour exécuter des instructions goto non locales (sortes d'exceptions).
<signal.h>	Pour contrôler les *signaux* (conditions exceptionnelles demandant un traitement immédiat, par exemple signal de l'utilisateur).
<stdarg.h>	Pour créer des fonctions avec un nombre variable d'arguments.
<stddef.h>	Définit plusieurs types et macros utiles, comme NULL.
<stdio.h>	Fournit les capacités centrales d'entrée/sortie du langage C, comme la fonction printf.
<stdlib.h>	Pour exécuter diverses opérations dont la conversion, la génération de nombres pseudo-aléatoires, l'allocation de mémoire, le contrôle de processus, la gestion de l'environnement et des signaux, la recherche et le tri.
<string.h>	Pour manipuler les chaînes de caractères.
<time.h>	Pour convertir entre différents formats de date et d'heure.
<stdint.h>	Déclare tous les types d'entiers (C99)
<stdbool.h>	Déclare le type boolean (C99)