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

HTA-FR – Kurs Telekommunikation

Verfasser:
D. Gachet / HTA-FR - Telekommunikation

Embedded systems 1 und 2

Programmiersprache C - Einführung

Klasse T-2 // 2018-2019

Inhalt

- Einführung
- Geschichte
- Merkmale der Sprache
- Werkzeuge
- Syntax
- Standard-Bibliotheken

Was ist ein Programm?

- **Gesamtheit der Tätigkeiten im Zusammenhang mit der Definition, dem Schreiben, der Entwicklung und der Ausführung von Informatikprogrammen; Sequenz von Befehlen, denen eine Einrichtung gehorchen muss.**

Nach Dictionnaire Larousse online (http://www.larousse.fr/dictionnaires/francais/interruption)

- **Ein Informatikprogramm ist ein Algorithmus, der aus 2 Grundelementen besteht:**
  1. Den **Aktionen**, die vor ihrer Ausführung durch eine Folge von **Befehlen** beschrieben werden
  2. Den **Daten**, die durch **Variablen** dargestellt und während der Befehlsausführung manipuliert werden

Inhalt des C-Kurses

- **1. Etappe**
  - Einführung
  - Basiselemente der Sprache
  - Komplexe Strukturen
  - Zeiger und dynamische Speicher
  - Funktionszeiger
- **2. Etappe**
  - Funktionen und Schnittstellen mit dem Assembler
  - Eingänge/Ausgänge
- **3. Etappe**
  - OO-Design
- **4. Etappe**
  - Entwicklungswerkzeugkette
  - Einheitliche Tests und Behandlung des Codes
  - Dokumentation

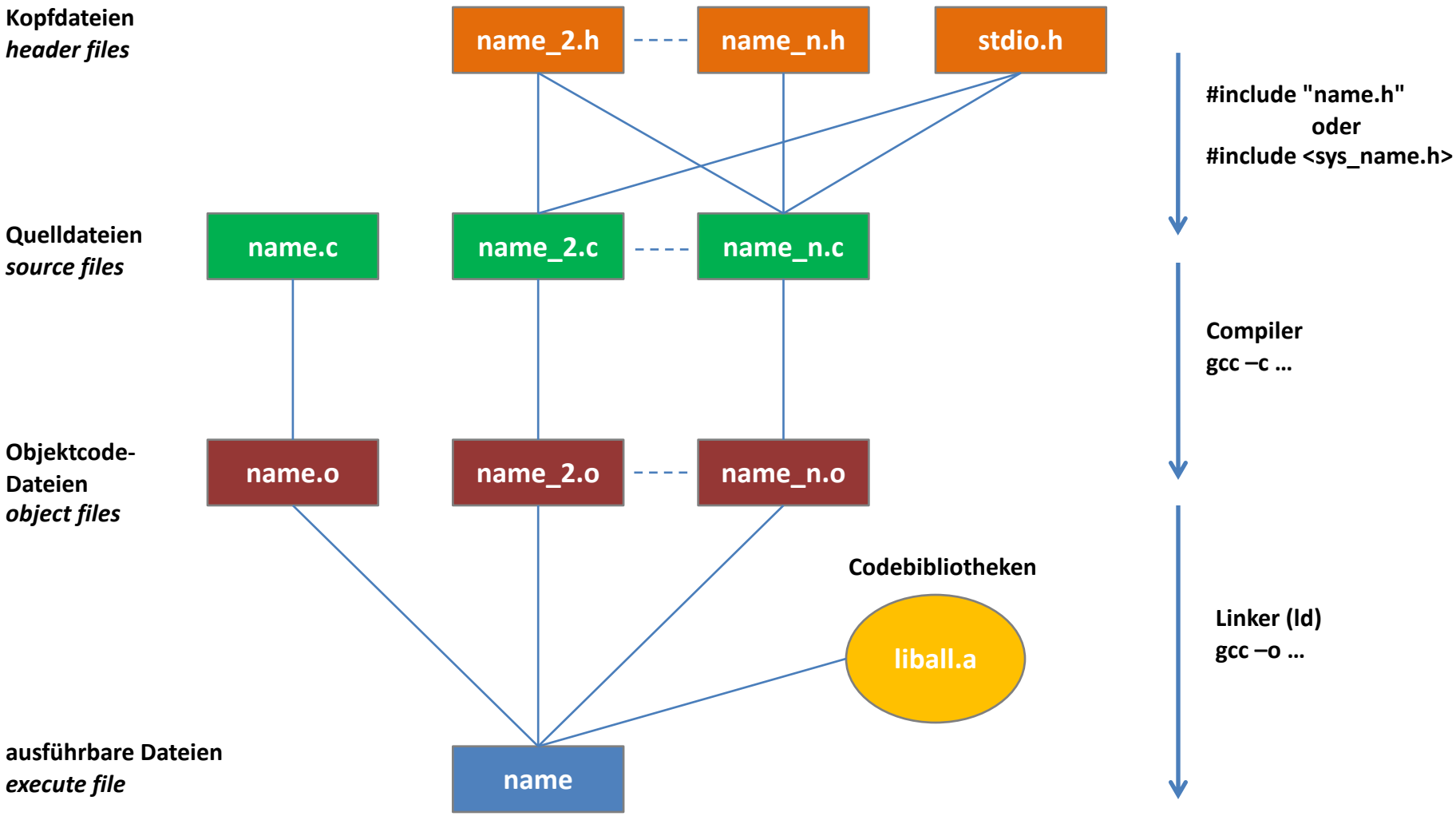
Geschichte von C

~1970	Dennis Ritchie und Kenneth Thompson entscheiden, zur Sprache B Verbesserungen hinzuzufügen und eine neue Sprache B (NB) zu kreieren
1972	Diese Verbesserungen der Sprache sind so gross, dass Ritchie entscheidet, diese neue Sprache Czu taufen
1973	90 % von UNIX werden in C geschrieben
1974	Die Bell Laboratorien vergeben UNIX-Lizenzen an Universitäten und verbreiten auf diese Weise die Programmiersprache C
1978	Brian Kernighan und Dennis Ritchieformalisieren die Sprache, die heute K&R C genannt wird, und schreiben ihr erstes Buch "The C Programming Language"
1983	Bjarne Stroustrup fügt ab Beginn der 80er-Jahre das Konzept der Klassen zu C hinzu und kreiert 1983 C++
1989	Standardisierung der Sprache und Kreation von ANSI C (C89)
1999	Überarbeitung der Sprache (C99)
2011	Letzte Entwicklung der Sprache (C11), früher (C1X)

Merkmale...

- **Nicht sehr hoch entwickelte Sprache, die aber eine offene Programmierung bietet, die die Manipulation von elementaren Objekten (Bit, Oktette, ...) und den Zugriff auf die Hardware ermöglicht.**
- **Die Sprache verfügt über keine Befehle für die Ein-/Ausgabe, die Speicherverwaltung, das Kommunikations- und Prozessverwaltungssystem. Alle diese Mechanismen werden durch das System mithilfe von Bibliotheken für Funktionen und Betriebssystemaufrufe bereit gestellt.**
- **Stärke der Sprache:**
  - Ausführungsgeschwindigkeit (optimale Kompilierung)
  - Leistungsfähige Operatoren für ganze Zahlen
  - Umfassende Verwendung von Zeigern
  - Erleichterter Zugriff auf die Hardware
- **Normalisierung der Sprache:**
  - Kernighan & Ritchie C, einfache Version der Sprache
  - ANSI-C, Standardisierung Prototypentwicklung, Normalisierung der Standard Bibliotheken
  - POSIX, Normalisierung der relevanten Betriebssystemfunktionen

Dateiorganisation unter C



Installation der GNU-Werkzeugkette + Eclipse

- **Unter Linux ist die Werkzeugkette als Vorgabe installiert**

- **Unter Mac OS X**
  - Installieren Sie die "command line tools" mithilfe von Xcode

- **Unter Windows:**
  - MinGW - GNU-Werkzeugkette für Windows:
    - http://www.mingw.org/
  - GDB – GNU-Debugger für Windows:
    - http://sourceforge.net/projects/mingw/files/GNU Source-Level Debugger/

- **Eclipse CDT:**
  - http://www.eclipse.org/

C-Programme weisen im Allgemeinen folgende Struktur auf:

```
/* Einbeziehung (#include <header.h>) von Deklarationsdateien (header files)
   für die Funktionen, Makros oder externe Variablen, die in andern
   Quelldateien definiert sind.
*/

// Definition der Datentypen
// Definition der globalen Variablen
// Deklaration der lokalen Funktionen im Kompilierungsmodul
// Definition der lokalen und globalen Funktionen

int main ()
{
  // Definition der lokalen Variablen im Block
  // Befehle, die den Programmalgorithmus implementieren
  return 0;
}
```

Die Funktion `int main()` bildet den Kern der Anwendung. Es ist die erste Funktion, die beim Starten einer Anwendung aufgerufen wird.

Kommentare

Jedes gute Programm muss dokumentiert werden. Die einfachste und effizienteste Form ist die Dokumentation direkt innerhalb des Codes mithilfe von Kommentaren.

Alle C-Compiler erkennen die Kommentarblöcke Diese werden aus Zeichenfolgen gebildet, die zwischen `/*` und `*/` eingeschlossen sind, zum Beispiel:

```
/*
 * dies ist eine kleine Kommentarzeile.
 */
```

Es ist wichtig, sich zu merken, dass Kommentare nicht verschachtelt werden können. Das erste Auftreten der Zeichenfolge `*/` beendet den Kommentar.

Gewisse neuere Compiler verstehen auch die Zeilenkommentare. Diese beginnen mit einem doppelten Schrägstrich `//` und enden am Zeilenende,
zum Beispiel:

```
i++; // der Index wird mit 1 inkrementiert
```

Die Programmiersprache C setzt sich aus den folgenden 33 Schlüsselwörtern (keywords) zusammen, die nachstehend alphabetisch aufgeführt sind:

```
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
```

Die Sprache definiert die folgenden Anweisungen

- **Einbeziehen von Dateien**
  `#include <path-spec>` → Standard Dateien
  `#include "path-spec"` → spezifische Dateien
- **Definition von Makros, die es erlauben, Text zu substituieren**
  `#define macro_identifier substution-text`
  `#undef macro_identifier` → um die Definition eines Makros zu vermeiden
- **Definition von Makrofunktionen**
  `#define macro_fonction_id([argument-list]) substution-text`
- `#define print(x)  printf("x=%d\n", x)`
- **Bedingte Compilierung**

```
#ifdef symbol
#ifndef symbol

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

Anweisungen (II)

- **Operatoren**

```
#    → wandelt den Parameter eines Makros in einen String um
##   → erzeugt aus 2 anderen Token ein Token

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

- **Vordefinierte Makros**

```
__FILE__     : zeigt den Namen der Quelldatei an.
__LINE__     : zeigt die erreichte Zeilennummer an
__DATE__     : zeigt das Kompilierungsdatum des Quellcodes an
__TIME__     : zeigt die Uhrzeit der Kompilierung des Quellcodes an
__VA_ARGS__  : Liste der variablen Argumente eines Makros (variadic macro)
```

- **Schutz für die Kopfdateien (guard for header files)**

```
#ifndef symbol          /* → zum Beispiel FILENAME_H */
#define symbol

#endif
```

Dieser Schutz basierend auf der Definition eines Symbols kann vorzugsweise durch das Pragma "once" ersetzt werden; es ist jedoch am sinnvollsten, beide zu kombinieren.

```
#pragma once
```

Beispiel:

```c
/**
 * A simple example 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;
}
```

Standard-Bibliothek (ANSI-C)

<assert.h>	Enthält das Makro assert, das als Hilfe dient, um inkohärente Daten oder andere Programmierfehler in einer Debugger-Version eines Programms festzustellen.
<ctype.h>	Funktionen, die für eine schnelle Klassifizierung der Zeichen oder für die Umwandlung von Gross- und Kleinbuchstaben unabhängig vom verwendeten Zeichencodierungssystem (character set) benutzt wird.
<errno.h>	Menge (oder öfter Teilmenge) der Fehlercodes, die durch die Funktionen der Standard Bibliothek über die Variable errno zurückgegeben werden.
<float.h>	Enthält Konstanten, die die von der Implementierung abhängigen Eigenschaften der Gleitkommazahlen spezifizieren, wie die minimale Differenz zwischen zwei Gleitkommazahlen, die maximale Anzahl der genauen Nachkommastellen und das Intervall der Zahlen, die dargestellt werden können.
<limits.h>	Enthält Konstanten, die die von der Implementierung abhängigen Eigenschaften der Ganzzahl-Typen (Integer) spezifizieren, wie das Intervall der Zahlen, die dargestellt werden können.
<locale.h>	Für die Anpassung an unterschiedliche kulturelle Konventionen.
<math.h>	Für die Berechnung häufig vorkommender mathematischer Funktionen.

Standard Bibliothek (ANSI-C) (II)

<setjmp.h>	Für die Ausführung von nicht lokalen goto-Befehlen (Ausnahmearten).
<signal.h>	Für die Steuerung der *Signale* (Ausnahmebedingungen, die eine unmittelbare Behandlung erfordern, zum Beispiel ein Benutzersignal).
<stdarg.h>	Für das Erzeugen von Funktionen mit einer variablen Anzahl Argumenten.
<stddef.h>	Definiert verschiedene Typen und nützliche Makros wie NULL.
<stdio.h>	Liefert die zentralen Kapazitäten die Ein- und Ausgabe der Sprache C wie printf.
<stdlib.h>	Für die Ausführung verschiedener Operationen, darunter die Umwandlung, die Erzeugung von Pseudo-Zufallszahlen, die Speicherzuteilung, die Prozesskontrolle, die Verwaltung der Umgebung und der Signale, die Suche und das Sortieren.
<string.h>	Für die Manipulation der Zeichenketten.
<time.h>	Für die Umwandlung zwischen verschiedenen Datums- und Uhrzeitformaten.
<stdint.h>	Deklariert alle Ganzzahl-Typen (Integer) (C99)
<stdbool.h>	Deklariert den Typ boolean (C99)