NTNU – Trondheim
Norwegian University of Science and Technology

Introduction to C
TDT4258 Energy Efficient Computer Design Lab

Stefano Nichele
Department of Computer and Information Science
2013, February 15th

Plan

- Basics:
  - The first program
  - Operators and flow control
  - Variables and datatypes
- Arrays and pointers
  - Arrays
  - Pointers
  - Array pointers and vice versa
  - Strings
- Functions
  - Declaration
  - Main
- Miscellaneous
  - Preprocessor
  - Header files
  - Standard library

The first program

The only way to learn a new programming language is by writing programs in it. The first program to write is the same for all languages:

Print the words

hello, world

Kernighan & Ritchie

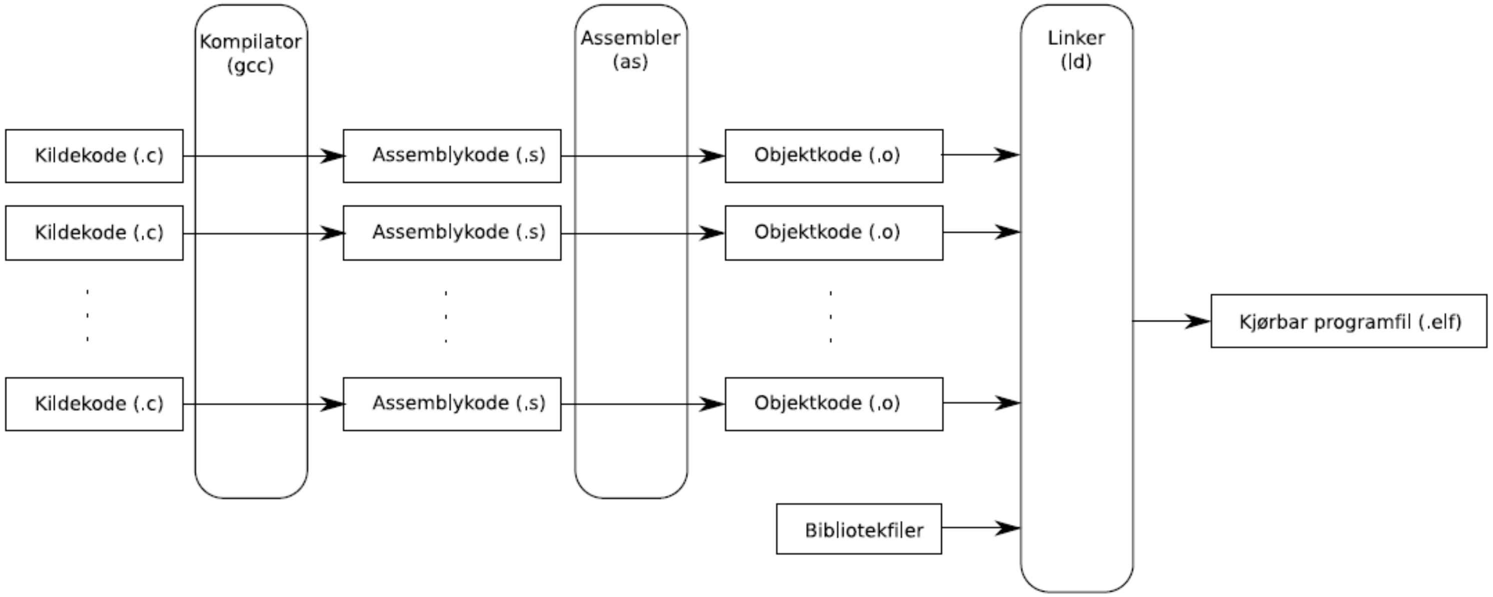
The first program

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

Compile and run:

```
$ gcc -o hello hello.c
$ ./hello
hello, world
```

Separate compilation



NTNU – Trondheim
Norwegian University of Science and Technology

Operators

- Arithmetical, logical, assignement and comparison operators:
  - Bitwise operators:

&	AND
|	OR
^	XOR
<<	left shift
>>	right shift
~	negation

Flow control

- Same as Java
  - for
  - while
  - do while
  - if / else
  - switch

Data types

- Basic data types: char, int, float, double
- Variants: short / long, signed / unsigned
- Example:

  short int

  unsigned char

  unsigned long int
- Boolean values: use int (0 false, 1 true)
- Strings: use array of char that ends with '\0'

Modifications

- const: constan, value cannot be changed
- static (in function): the variable retain its value between each time the function is called
- static (on a global variable): the variable is local to the c file where it is declared
- extern: the declaration of the variable is in another file
- volatile: specifies that the variable should not be optimized

Struct and typedef

```
struct foobar {
int a;
double b;
char c;
};
struct foobar f1;
f.a = 5;
typedef struct foobar foobar_t;
foobar_t f2;
```

NTNU – Trondheim
Norwegian University of
Science and Technology

Arrays

- Arrays have a constant size
- The limits of the array are not checked automatically
- Example

```
int tab[5];
tab[0] = 7;
tab[4] = 8;
tab[5] = 9; /* error, but legal */
```

NTNU – Trondheim
Norwegian University of
Science and Technology

Pointers

- Pointer: a variable that contains a memory address
- Declare the type of variable that has to be pointed and an asterisk (*)
- & operator gives the address of the variable
- * operator dereferences a pointer (provides the content of the memory location it points to)
- void pointers can point to anything

Pointers - example

```
int a = 15;
int b = 24;
int *p1 = &a; /* p1 points to a */
int *p2 = &b; /* p2 points to b */
*p1 = *p2 + 1; /* equivalent to a = b + 1 */
p2 = p1; /* now pointer p2 also points to a */
```

Pointers – a realistic example

```
void swap(int *a, int *b)
{
int t = *a;
*a = *b;
*b = t;
}
```

Pointers - arithmetic

- You can get a new pointer by adding a pointer to an integer
- Example:

```
int tab[10];
int *p = &tab[0]; /* pointer to the first element */
p = p + 3; /* now p points to tab[3] */
*(p - 1) = 42; /* we set the value of tab[2] */
```

- Note that p+3 not necessarly increases the address by 3, but with 3s, where s is the size of an int

Array pointers and vice versa

- An array variable is really just a pointer to the first element in the array

```
int tab[10];
*(tab+3) = 5; /* same as tab[3] = 5 */
int *p = &tab[0];
p[3] = 7; /* same as *(p+3) = 7 */
```

NTNU – Trondheim
Norwegian University of
Science and Technology

Strings

- A text string is represented as an array of char values
- Ends with the magic value `\0`
- Test in double quote is automatically filled in the array:

```
char astr[] = "hello"; /* astr has length 6 */
```

NTNU – Trondheim
Norwegian University of
Science and Technology

Strings – example

```
/*
* Calculatea the length of the string
* (without '\0’)
*/
int strlen(char *str)
{
int n = 0;
while (str[n] != '\0')
n++;
return n;
}
```

NTNU – Trondheim
Norwegian University of
Science and Technology

Return type and parameters

- Same as Java
- Use (void) as a parameter if the function should not take arguments

Prototypes

- A function cannot be called before it is declared
- Prototype: specifies just the name, return type and parameters, not the code

```
void swap(int *a, int *b);
```

Main

- int main(int argc, char **argv)
- Returns 0 if everything goes well
- argc: number of command line arguments
- argv: the command line arguments

Main - example

```
/*
* Program that writes
* the command line arguments
*/
#include <stdio.h>
int main(int argc, char **argv)
{
int i;
printf("%d arguments\n", argc);
for (i = 0; i < argc; i++)
printf("%d: %s\n", i, argv[i]);
return 0;
}
```

NTNU – Trondheim
Norwegian University of Science and Technology

Preprocessing

- A separate step before the actual compilation
- Make simple modifications in the source code
- The most important directives are #define and #include

#define

- Defines a constant or a macro
- Example
  - #define ANSWER 42
  - #define sq(x) ((x)*(x))
- Proprocessing now replaces all the occurrences of ANSWER with 42 and sq(x) with (x)*(x) (for all x)
- Example:
- From sq(ANSWER+1) to ((42+1)*(42+1))

#include

- Includes a file

#include <filename>: System files
(example: #include <stdio.h>)

#include ”filename": Local file
(example #include "foobar.h”)

Header files

- They contain function prototypes, struct, definitions, preprocessing directives, external declarations
- They do not contain variable definitions and functions (the code)

Standard library

- Some useful libraries:
  - #stdio.h: printf, scanf, file I/O
  - #string.h: string functions, copy, comparison
  - #math.h: trigonometric functions, logarithm etc.
- Every standard library has its onw man-page in section 3, for example man 3 printf